NPS52-89-037

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

THE COMMAND AND CONTROL WORKSTATION OF THE FUTURE

Subsurface and Periscope Views

Gordon Kenneth Weeks, Jr.

Charles Edward Phillips, Jr.

June 1989

Thesis Advisor: Michael J. Zyda

Approved for public release; distribution is unlimited

Prepared for:

Naval Ocean Systems Center
Code 402
San Diego, CA 9215

Naval Underwater Systems Center
Combat Control Systems Department
Building 1171/1
Newport, RI 02841

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral R. C. Austin
Superintendent

Harrison Shull
Provost

This report was prepared in conjunction with research conducted for the United States Naval Underwater Systems Center, Newport, Rhode Island and United States Naval Ocean Systems Center, San Diego, California. The work was funded by the Naval Postgraduate School.

This thesis is issued as a technical report with the concurrence of:

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS None
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Unlimited Distribution
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) NPS52-89-037	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable) 368	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION NPS 8b OFFICE SYMBOL (If applicable) 37/368	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER Direct Funding
8c ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification) THE COMMAND AND CONTROL WORKSTATION OF THE FUTURE SUBSURFACE AND PERSICOPE VIEWS(U)

12 PERSONAL AUTHOR(S) Phillips Jr., Charles, E., CPT, USA
Weeks Jr., Gordon, K., LT, USCG

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	June 1989	111

16 SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of the Defense or the U. S. Government.

17 COSATI CODES		
FIELD	GROUP	SUB-GROUP

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
High Performance Graphics Workstation, Real-time 3D Graphics, Subsurface View, Periscope View, 3D Terrain Visualization, 3D Icons, CCWF

19 ABSTRACT (Continue on reverse if necessary and identify by block number)
Today's tactical commander needs to assimilate enormous amounts of information to make reasonable decisions. Graphics displays can be a valuable tool in conveying such information in a clear and concise manner. Our vehicle for studying such displays is a project entitled the Command and Control Workstation of the Future (CCWF). In this study, we focus on the subsurface and periscope views. Our primary goal is to provide the tactical commander with an interactive, user-friendly, 3D simulation system to assist in decision making, planning and training. The secondary goal is to provide information on real-time three dimensional graphics techniques, with an eye on meaningful graphics workstation performance measurements for such techniques.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL Micheal J. Zyda	22b TELEPHONE (Include Area Code) (408)646-2305 22c OFFICE SYMBOL 368

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

GPO: 1986-606-24

Approved for public release; distribution is unlimited.

THE COMMAND AND CONTROL WORKSTATION OF THE FUTURE: SUBSURFACE AND PERISCOPE VIEWS

by

Gordon Kenneth Weeks Jr.
Lieutenant, United States Coast Guard
B.S., United States Coast Guard Academy, 1984

and

Charles Edward Phillips Jr.
Captain, United States Army
B.S., United States Military Academy, 1981

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1989

ABSTRACT

Today's tactical commander needs to assimilate enormous amounts of information to make reasonable decisions. Graphics displays can be a valuable tool in conveying such information in a clear and concise manner. Our vehicle for studying such displays is a project entitled the Command and Control Workstation of the Future (CCWF). In this study, we focus on the subsurface and periscope views. Our primary goal is to provide the tactical commander with an interactive, user-friendly, 3D simulation system to assist in decision making, planning and training. The secondary goal is to provide information on real-time three dimensional graphics techniques, with an eye on meaningful graphics workstation performance measurements for such techniques.

Captain Phillips designed and built the 3D vessel and weapon icons, was responsible for the layout of the initial 2D and 3D windows, and was a major force in the current standard file format work by providing vessel models in both formats and giving realistic operational guidance on its use. Lieutenant Weeks provided the overall program design including considerations for future expansion. He created and implemented the 3D operations, 2D operations and networking scheme. His major focus was in creating the high performance terrain drawing algorithms using standard DMA terrain databases and ensuring the overall programs ease of modification.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

Since this project is the initial work on the subsurface and periscope views of the Command and Control Workstation of the Future, all code generated by this project is original with the exceptions noted below. We started with the concept of drawing the terrain in a way different from any other project here at NPS. Performance tests would seem to show we are successful.

We would like to thank the following people for their timely help. Captain Randolph Strong, USA, provided a copy of his low level networking routines which were modified and used as the low level building blocks for the networking scheme used within the CCWF Subsurface and Periscope Views [Strong, 1989].

Captain Emil Velez, USA, provided the initial programs that converted 3D vessels from the standard IRIS format to the new standard file format. Upon his departure in March, 1989, Captain Phillips became the sole source for converting the current 3D vessel icons to the standard file format, using up considerable amounts of time.

Finally, we would like to thank our thesis advisor, Dr. Michael Zyda, for his guidance, graphics techniques, and the occasional ear that helped a lot of steam get blown off without hurting anyone.

I. FOCUS

Our project is the preliminary work on the Command and Control Workstation of the Future (CCWF): Subsurface and Periscope Views (S&PV). The view considerations are much different for an underwater view over a surface view and are in fact entirely synthetic. While there is normally only a restricted underwater view, the concepts used in previous projects can be considered when developing the view. The goal of this project is to produce a visualization tool that provides a three dimensional perspective of an underwater area of interest. This view is an estimate of what would be seen from the bridge or periscope of a submarine if the view was not limited by darkness or underwater visibility. Untying the commander from standard sonar readings and two dimensional navigational charts should give the commander a better feel for the operations area. This visualization tool can be a valuable navigational aid, with actual terrain data being displayed graphically. Previously unseen terrain features can be used to aid the tactical commander or commanding officer in best applying his assets. The simulation can further be used to navigate suggested vessel tracks before actually maneuvering in the area. Many avenues of approach can be attempted ahead of time helping to establish the most advantageous route.

A. THREE DIMENSIONAL DISPLAY

The three dimensional display is the most important part of this project. With proper lighting and shading, the tactical commander can get a strong feel for his present underwater environment. Along with the 3D terrain, vessels in the vicinity

can be portrayed in their current operating environment. Seeing these vessels can be a valuable aid in making tactical decisions based on the spatial relationships to those vessels and their associated terrain. When confronted with an enemy vessel, a commander can make tactical decisions based on what terrain can provide him with the most cover and concealment and still leave room for possible evasive action. Assimilating three dimensional data from the traditional two dimensional command and control system in the heat of battle can easily become error prone resulting in possible tactical blunders.

The S&PV presents the subsurface commander with iconic 3D exterior hull silhouettes of surface vessels. These silhouettes show the submarine commander the enemy formations he is up against allowing him to develop his plan of attack and any counteractions that may be necessary. The tactical commander is given the flexibility to select an area of operation, and the type of vessel he wishes to use. Such capabilities provide valuable training for learning how best to fight a particular platform. Each vessel type, with its unique operating characteristics, reacts differently to its current tactical environment. The tactical commander is made familiar with each vessel's operating characteristics. Moving the commander's viewpoint to different vessels enables the commander to see what his own vessel looks like from the eyes of the enemy and learn what can be done to improve his situation.

B. SUMMARY OF THE CHAPTERS

Our work on the subsurface and periscope views for the CCWF is a fairly large effort. In Chapter 2, we begin by discussing the history of real time visual simulators at the Naval Postgraduate School (NPS). In Chapter 3, we discuss real time

graphics hardware and techniques. We primarily focus on the Silicon Graphics IRIS 4D/ 70GT and its supporting software packages. Chapter 4 provides an overview of the S&PV from its DMA databases to networking operations. Chapter 5 is concerned with the 2D operations within the S&PV. These functions include viewing and picking a sixty nautical mile square data file, placing vessels within the selected area and finally picking a vessel to navigate. Chapter 6 focuses on 3D operations within the S&PV. Network operations are detailed in Chapter 7 while creating the iconic 3D exterior hull silhouettes is the subject of Chapter 8. Chapter 9 contains performance evaluation information on the S&PV system. Chapter 10 contains conclusions, limitations and future directions.

II. REAL -TIME VISUAL SIMULATORS AT NPS

A. INTRODUCTION

The Naval Postgraduate School's Graphics and Video Laboratory has had a long history in developing real-time 3D visual simulators [Ref. 3,4]. The current version of the Moving Platform Simulator (MPS) is the fruit of nearly three years of effort and the summation of two separate systems [Ref 3:pp 4-5]. It is currently being developed in three separate flavors which in the long run will be merged together.

The Command and Control Workstation of the Future (CCWF) project was initiated to provide the tactical commander with ability to view a scenario from any one of his platforms. This capability enhances the commander's abilities to maneuver and if necessary, fight the vessels under his control. He is better able to determine each vessel's tactical environment by positioning himself on the vessel. The initial start of the CCWF was with the surface view simulator and Commanders Display System [Ref 1,4]. We discuss these two systems as they relate to the Subsurface and Periscope View Simulator (S&PV).

B. COMMANDERS DISPLAY SYSTEM

The Commanders Display System (CDS) was the original part of the Command and Control Workstation of the Future project [Ref.1: p. 18]. It was to provide the contact and tactical data for all of the simulator view programs. First, the program's user selects the platform he desires to see. Then, the area of interest, as seen from

the vessel, is displayed by the appropriate 3D view program. The CDS was written on the IRIS 2400 Turbo and currently has not been ported to the IRIS 4D/70GT. A port of the CDS to the newer IRIS machines is a future effort [Ref. 1:p. 18].

An important consideration in the CDS was its use of Navy Tactical Display System (NTDS) fonts to display the various contacts. These fonts were color coded as to the intention of the contacts and deployed as the contact locater [Ref. 1:pp. 18-24]. The carry over from the real tactical environment to the simulator is important in the use of our simulators as training tools.

C. SURFACE VIEW SIMULATOR

The surface view simulator was designed to display a visible area of operation as seen from a surface ship or helicopter [Ref 4:pp. 43-53]. It featured visibility ranges of up to twenty six nautical miles. It used the same databases as used by the subsurface and periscope view but never included many features that are currently in our simulator such as lighting, networking and vessels. It did have the capability of loading four databases at once to provide for the twenty six mile visibility range [Ref. 4:pp. 43-53].

The surface view simulator used multiple resolution terrain displays. Its high resolution area was displayed with 100 yard data points. The medium resolution areas were displayed with 1,200 yard data points and finally, the low resolution data was done with 12,000 yard data points. The multiple resolutions were used to allow the simulator to display the 52,000 yard, twenty six nautical mile viewing areas desired of the project in real-time [Ref. 4:pp. 49-51].

The Surface View Simulator was a first prototype. Work on that system has been stopped. That system is superseded by the Subsurface and Periscope View simulator.

D. SUBSURFACE AND PERISCOPE VIEWS

The S&PV is the latest work in the CCWF project. It provides an effective simulated subsurface view with variable visibility ranges and a limited surface view to 20,000 yards. The S&PV contains a realistic lighting model and 3D vessel icons not found in the surface view simulator. It is a very stable simulator that can be enhanced to include a complete surface and air view without the instability found in the earlier, pioneering projects.

III. REAL-TIME GRAPHICS TECHNIQUES AND HARDWARE

The Silicon Graphics Inc., IRIS 4D/70GT is a powerful, fairly low cost color graphics workstation. The IRIS provides a screen resolution of 1280 by 1024 pixels with 96 bit planes per pixel. These 96 bit planes provide double buffered, alpha buffered, and z-buffered displays with overlay and underlay capabilities. The IRIS has a proprietary graphics pipeline which provides hardware and microcode support for Gouraud shaded polygon fill and lighting in real time. With the rated speed of 40,000 Gouraud shaded, lighted, four sided polygons per second, the IRIS 4D/70GT provides a good platform on which to base the S&PV simulator [Ref. 2].

A. COLOR

Only recently has it become cost effective to use color in graphics displays. Color can be used as a very powerful visual cue in depicting various types of information. In the past, the commander had to rely on the expertise of the operator to interpret the information on the screen. With color, the commander can more easily differentiate enemy and friendly units, increasing his ability to make accurate decisions quickly. Chart data can be displayed using realistic color shading to provide for efficient maneuver plotting. Vessel icons can be overlaid onto the conning charts and used to master the tactical environment in which the tactical commander or commanding officer is currently involved.

B. GOURAUD SHADING

Gouraud shading is provided in hardware by the IRIS 4D/70GT. This shading technique provides for realistic shading of polygons with graduations of color. While a less sophisticated fill algorithm would shade the polygon with the last color set, Gouraud shading linearly shades the polygon between vertex color points. With Gouraud shading, a four sided polygon can have a different color set at each vertex and the shading blends the colors into a smoothly shaded polygon. All polygons in the S&PV simulator are Gouraud shaded [Ref. 5:pp. 289-291].

C. LIGHTING

Another significant feature of the IRIS is its hardware support for efficiently performing lighting calculations. Proper lighting enhances the realism of objects within the simulator. Three things are considered when using a lighting model, light sources, the objects to be lighted and the position of the eye. For a light source, the color, location and direction of the light source is required. Each object's color and surface composition is calculated along with the designated eye position in relation to the object and the light source. The combination of the these factors are used to create realistic, lighted scenes. Surfaces that are shielded from the light source tend to be darker and give the illusion of shading. Surfaces that are perpendicular to the light source are the brightest. The IRIS graphics pipeline calculates the proper color to depict objects using the angular relationships between the viewer, the object surface and the light source. Lighting requires extra work and data storage but the effects are dramatic as seen in the simulator.

D. OVERLAYS

Overlays are used in this project to save drawing time. Of the 96 bits per pixel, four are used to support overlay or underlay drawing. These bits can be cleared and drawn without effecting the drawing contained in the RGB, double buffered bits. The S&PV uses overlays to outline the five nautical mile piloting grid area on the sixty nautical mile database grid. The drawing time for the database grid is approximately 7 seconds, far too slow for redrawing during the real-time operations. The overlay planes are cleared and the piloting grid outline is updated as required without affecting the database grid.

E. DOUBLE BUFFERING

Double buffering is a graphics technique used for animation. Double buffering allows a drawing to be completed in the back buffer before it is displayed. Animation speeds are recorded in frames per second. To maintain a useful interactive simulation, drawing times are maintained at better than three frames per second.

F. Z-BUFFERING

Hidden surface removal using z-buffering is an important attribute of the IRIS. Hidden surfaces are surfaces that are obscured by other objects closer to the observer [Ref. 5:pp. 262-264]. The IRIS calculates which surfaces are closer to the observer on a pixel by pixel basis using the Z-buffer (or depth buffer) [Ref. 2]. This technique is particularly valuable in terrain drawing, which is the heart of the simulation.

IV. SEA OF JAPAN DATABASES

The subsurface and periscope views (S&PV) module of the CCWF was designed to display Defense Mapping Agency Sea of Japan terrain data in real-time. Along with the terrain, the S&PV displays 3D vessel icons and is capable of networking between many graphics workstations.

A. SEA OF JAPAN DMA DATABASE

Each of the Sea of Japan databases used in the CCWF covers an area of sixty square nautical miles. Each database file consists of 1201 by 1201 height values whose lower left hand corner is the whole degrees file name. For example, the file 31N130E has a lower left hand coordinate of 31 degrees north and 130 degrees east. Each data point is one hundred yards, three seconds of latitude, apart. From the starting point, the data points progress north every hundred yards until the north most point with the current longitude is found. The next point is then the south most point with the new longitude incremented 100 yards or 3 seconds.

Each height value is given in meters and is represented as a sixteen bit integer. Since underwater terrain data is classified, or simply not available, we flooded the databases we had 1,000 feet, 308 meters, to derive our underwater terrain. Original ocean points are indicated with deep blue which represents the bottom of the database. No vessels are allowed to progress deeper than this value.

V. TWO DIMENSIONAL OPERATIONS

The two dimensional operations within the S&PV are in two parts. The first step is selecting the desired database grid on which to operate. All currently available databases can be viewed before the final selection is made. Once the database has been selected, the terrain vertex normals are calculated.

The second part of the 2D operations is the vessel placement and selection operations. This part is broken down into three basic steps, vessel placement, vessel selection and finally, new operations area or new vessel selection. All of the steps in this phase can be reversed or altered as the needs of the user require.

A. SIXTY NAUTICAL MILE DATABASE SELECTION

After the initial greeting screen, the user is prompted to select a database name. Once the desired database has been selected, the program portrays the database upon the screen as seen in Figure 5.1. The program then prompts the user whether or not he wishes to use the database for the simulation. If the user does not want to use the database, or simply wants to see what the others look like, the user selects the "don't use this grid" option. The program then prompts the user with the name of the databases not currently visible and the user makes a new selection. This process continues until the user finally selects the "use this grid" option on the prompted menu.

Once the database has been selected, the program calculates the terrain vertex normals. Each of the 1,201 by 1,201 data points has a vertex normal associated with

Figure 5.1 CCWF 2D Screen

it. The selected grid is redrawn on the screen in the order of the normal calculation for those points drawn. This process lets the user know the program is working on an otherwise user invisible operation. Once the normals have been calculated, the selection grid and the main 2D operations run-time menu is popped up for the user.

B. VESSEL PLACEMENT

The main 2D operations control menu offers the user three options, vessel placement, vessel selection and retention and placement of the currently driven vessel. The controlling switch statement is shown in Figure 5.2. Vessel placement operations allow the user to select one of the five currently available platforms and place them anywhere on the sixty nautical mile database grid.

Immediately upon selecting the place vessel option, the program queries the user to select a five nautical mile grid on which to place the vessels. Once this has been accomplished, the user is queried for the vessel type he wishes to place. The following vessels are currently available, the Kitty Hawk class aircraft carrier, the Spruance class guided missile destroyer, the Ohio class ballistic missile submarine, the Los Angeles class fast attack submarine and the Russian Typhoon class ballistic missile submarine. Once the vessel type has been selected, the user is queried for the course, speed and, if a subsurface platform, the depth desired. Once all of the appropriate data has been selected from the menus, the user is directed to place the vessel by positioning the mouse on the location and pressing the left mouse button. Once the vessel has been placed, the user is asked if he desires to place more platforms on this grid square. The user can place as many vessels on the grid square as desired. Figure 5.3 shows the vessel creation and placement control routine.

```
while (picked_driven_vsl == FALSE)
{
  /* On the initial Pass, the 2D map is already there so draw the grid */

  if (initialPass == FALSE)
    drawSea2D(MAP2D,FALSE);
  else
  {
    setwindow(MAP2D);
    frontbuffer(TRUE);
    drawGridandVessels();
    frontbuffer(FALSE);
    initialPass = FALSE;
  }

  /* Discover what the user would do */

  switch(promptmenu(ControlMenu))
  {
    case 1 :  /* Place New Platform */
        placePlatform();
        break;

    case 2 :  /* Select Platform to drive */
        picked_driven_vsl = selectDrivePlatform();
        break;

    case 3 : /* Retain current platform */
        picked_driven_vsl = place_retained_vsl();
        break;

    default :
        break;
  } /* end switch */
} /* end while */
```

Figure 5.2 2D Main Control Routine

```
/* Select grid location */

selectGrid();

/* Loop til all desired vessels are placed */

placeanother = TRUE;
while (placeanother)
{
  /* Create new platform */

  new_vsl = add_vsl_packet();

  /* Select platform type */

  selectNewPlatform(new_vsl);

  /* Select course, speed and depth */

  selectCourseSpeedDepth(new_vsl);

  /* Place the vessel */

  putContactVsl(new_vsl);

  /* Find the users desires */

  if (promptmenu(placeMenu) == 2)
    placeanother = FALSE;
  else
    drawSelectGrid(MAP2D,(float)opareaX+50,(float)opareaY+50);
}
```

Figure 5.3 Vessel Placement Control Routine

Vessel courses are currently set for every thirty degrees starting at true north. Vessel speeds are from zero to five knots, then every five knots up to thirty knots. Subsurface vessels are allowed from draft depth to 300 meters below with fifty meter increments. Any increases in the above ranges can be easily accomplished by changing the appropriate menu creation and interpretation code.

Each vessel's location is displayed on the five nautical mile piloting grid and the sixty nautical mile database grid using the ACDS fonts taken from the Commander's Display System as seen in Figure 5.4 [Ref 1:pp. 18-24]. These fonts are further coded with colors appropriate to their supposed intentions. The color list ranges from blue for friendly forces to red for hostile forces. While the colors used within the program were set by us, coloring the icons by their intention was first done in the Commander's Display System [Ref 1:pp. 18-24]. Each vessel icon has a blue line emanating from the center of the icon indicating the selected course for the vessel.

When the user has placed all desired vessels within the selected five nautical mile grid, the program returns to the main 2D control menu, interpreted by the code in Figure 5.2. The user can again choose to place more vessels within the sixty nautical mile database grid by selecting the vessel placement option. There is no restriction keeping the user from re-selecting the same five nautical mile grid for placing vessels.

C. VESSEL SELECTION

Once the user has placed all desired vessels onto the sixty nautical mile database grid, the user then normally selects a vessel to drive. This is accomplished by first picking a five nautical mile grid. The program then prompts the user with instructions to place the mouse on the desired vessel icon and press the left mouse

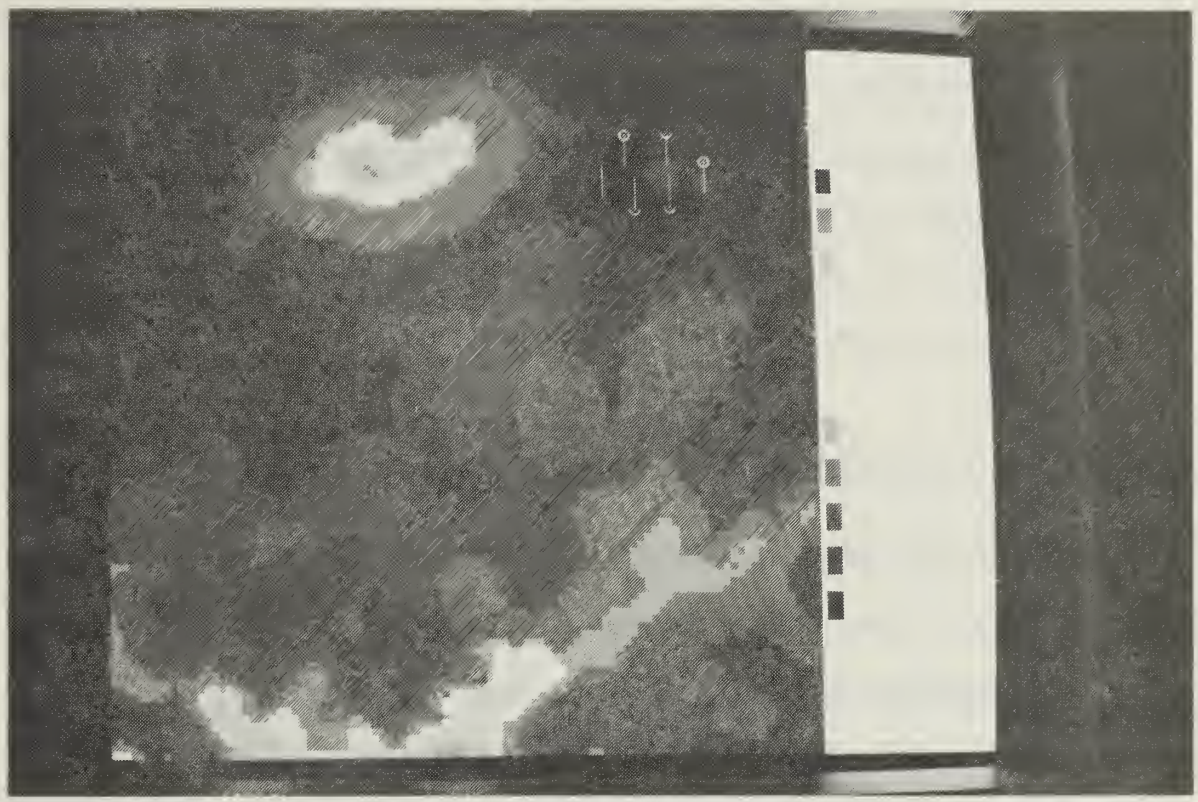


Figure 5.4 Five Nautical Mile Placement and Selection Grid

button. The screen blanks to black when the program has successfully found the desired vessel and then initializes the 3D screens and operations.

If the user had earlier selected the "restart with a new operation area" 3D run-time option, the user's vessel at the time of the selection would still be attached to the user. If the user desires to navigate a different type of vessel or see what the other vessels might be seeing, this 2D menu option allows for vessel swapping. Since the user is currently in a selected vessel, the program asks if the user wants to place his current vessel back with the other vessels or discard it from the simulation. Once this decision has been made, the program again queries the user for the desired five nautical mile grid and the desired vessel is selected as before.

D. RETAIN DRIVEN VESSEL

When the user merely wants to move the driven vessel to a new operation area, the user selects "restart with a new operation area" from the 3D run-time menu. This option returns the program to the 2D main control menu, Figure 5.2. Selecting the "retain and drive current platform" menu option allows the user to move his current vessel to anywhere in the sixty nautical mile database grid. Once selected the user is prompted to select a five nautical mile grid for final placement upon the grid. The program then immediately returns to the 3D operations.

If the user selects the retain option without first having selected a vessel to drive, the program warns the user of his error and returns him to the main 2D control menu. The same is done in the "select vessel to be driven" menu option if no vessels were placed.

E. EFFECTS OF THE 2D OPERATIONS

While the 2D operations are primarily concerned with creating and placing vessels and selecting a vessel to drive, the routines in the 2D operations add a considerable amount of flexibility to the overall program. Using these routines, the user can change platforms, change area of operations and delete platforms.

Changing platforms is easily done starting at the 3D run time menu. Selecting the "restart with new operations area" menu item returns the user to the 2D main control menu. Selecting the "select platform to drive" option allows the user to replace his current vessel into the vessel contact list and select a new platform to drive. The user is queried for the five nautical mile grid the desired vessel is in. Once in the grid, the user selects the vessel desired as in the normal selection process. If the discarded driven vessel is in the grid selected, the user can reselect the vessel if desired. This does have the side effect of reloading the weapons on the vessel.

Changing operations area is the primary reason for the third option on the 2D main menu, "retain and drive current platform." Using the procedure discussed above, the user can easily move around the entire sixty nautical mile database grid.

Deleting vessels can be done by selecting the vessel to be deleted as the driven vessel. Returning to the 2D operations via the restart option, the user then selects the "select platform to drive" 2D main menu option. Since the user is currently driving a vessel, the user is then prompted as to the disposition of the vessel currently driven. To delete the vessel from the database, simply select the "discard the vessel" menu option.

VI. THREE DIMENSIONAL SIMULATOR OPERATIONS

After all currently desired vessels have been placed and the driven vessel's initial position has been determined, the simulator closes all of the 2D operation windows and initializes the 3D operations. The overall design of the 3D view, Figure 6.1, is to portray all necessary information to safely pilot and if necessary, fight the vessel. The selected operation area location is shown on the sixty nautical mile database grid coinciding with the five nautical mile piloting chart. Both are used as a navigational aid. A visual depth gauge is used for grounding avoidance. A weapons panel and a control panel give the current vessel and simulator status. The centerpiece 3D view presents the user with a simulated visual scene along the selected view direction. All six of the windows are either redrawn or updated with each cycle through the simulator.

A. 3D OPERATION OVERVIEW AND TERM DESCRIPTION

The 3D view was the major focus of this real-time simulator. It has gone through many revisions to get to the current state. The three dimensional underwater terrain is drawn with two major considerations, unbounded moving view point and processing speed. Unbounded moving view point releases the user from being tied to one particular terrain section and allows the viewer to roam the entire database and beyond. Processing speed is the primary concern for a real-time system. Speed, as measured in frames per second, is the guideline for performance measurement within this program.

Figure 6.1 S&PV 3D Screen

1. 3D View Considerations

The view presented to the user is always centered at the driven vessel's position and radiated out according to the desired view direction and view angle. View direction is the direction of view in relation to true north. This direction is set via a dial on the IRIS button and dial box. Through a menu option, the view direction can be tied directly to the course the vessel is traveling. The simulated view is then equivalent to the view from the bridge looking directly forward. The view angle is changed through a menu option on the main run time menu and can be changed from a forty five degree normal view to a fifteen degree zoom view. This feature controls the width of the base of the viewing triangle allowing the perspective command to either enlarge the visible objects in the zoom view or keep them at their normal perspective with the forty five degree angle.

A second menu option directly effecting the 3D view is the viewing distance. Currently, visibility ranges start at 5,000 yards and extend to 20,000 yards by 2,500 yard increments. The default value of the simulator is 7,500 yards.

2. Processing Speed

Processing speed of the simulator is a constant concern. All expensive calculations, such as sine and cosine values are stored for reuse. Frames per second, the speed at which the IRIS updates the view screen, is averaged over ten updates. Frames per second readings of less than 3 frames per second are considered too slow for real time use. Scenes below this threshold become very jerky and hard to control precisely. Since the simulation involves six active windows, a lighting model, ship

and submarine models and networking, streamlining code to the graphics display hardware is given high priority in code generation.

B. 3D PROCESSING SCHEDULE

The overall control of the three dimensional display is directed from the main program control and display loop, Figure 6.2. One iteration through the loop involves testing the graphics queue, updating all vessel positions, checking the network and processing packets as necessary, and finally redrawing and updating all of the simulation windows. This display and control loop is imbedded within the restart with a new operation area loop, Figure 6.3.

C. EVENT QUEUE PROCESSING

The graphics queue is the first thing checked within the control and display loop of Figure 6.2. Items queued allow the program to change the simulation environment. All entries on the queue are processed before moving to the update of vessel positions routine. Changes performed by processing the main run-time menu include tying the course and view, stealthing the submarine, changing the viewing distance and view angle, restarting with a new operational area, changing current terrain drawing method, turning off or on the lighting model, initiating or terminating network operations and ending the submarine simulator. The interaction of these options is discussed below in their area of influence within the drawing algorithms.

The graphics queue also interprets the IRIS button and dial box used by the S&PV. The bottom four dials are used to change the vessel's course, speed, depth and view direction. Figure 6.4 shows which dial relates to which event. The button

```
/* 3-D Simulation Control and Display Loop */

CONTINUE = TRUE;
while(CONTINUE)  /* Display Loop */
{
   /* While something is on the control queue */

   while (qtest())
   {
     CONTINUE = processQtoken(reStart);
   }

   /* Send Q changes to the network */

   sendchanges();

   /* Update all vessels according to the time expended and their */
   /* course and speed.                                           */

   updatepositions();

   /* Check network */

   if (Networking)
     checkNetwork();

   /* Redraw the 3D screens with the new views */

   redraw3Dscreens();

   /* Check for the grounding incident */

   if (CONTINUE)
     CONTINUE = bottomKill(reStart);
} /* while CONTINUE */
```

Figure 6.2 Main Simulation Control and Display Loop

```
/* Restart With New Oparea Loop */

while (*reStart)
{
    *reStart = FALSE;

    /* Perform the 2D selection and placement operations */

    draw2Dcontrol();

    /* Initialize 3D map and dials, switches and mouse */

    initialize3D();
    initialize_evaluator();

    /* Perform initial 3D draw routines */

    draw3Dcontrol();

    /* Set the device control window  */

    setwindow(CONTROL);

    /* Start real time loop processing and timing */

    startTimer();
    startFrameMeter();

    /* 3-D Simulation Control and Display Loop */
    /* See Figure 6.1                          */

  if (*reStart)
      initializeReStart();
} /* while reStart */
```

Figure 6.3 Restart With New Oparea Loop

box is used to fire the four weapons each vessel is allowed to have. We use the right most four vertical buttons.

1. Sending Changes to the Network

When the networking mode has been selected, changes caused by input device processing are sent to the network upon completion of the queue processing. Sending updates from within the queue processing routine resulted in an excessive number of updates being sent to the network when the dial devices were updated. A flag within the queue processing file is set if changes were made and the end result is sent to the network. Figure 6.2 shows the position of the sendchanges() routine.

D. VESSEL AND WEAPON POSITION UPDATING

Updating the vessel and weapons positions requires first calculating the time expended on the current pass through the program. Each vessel and weapons position is then updated according to its current course and speed. All vessels and weapons including the network vessels and weapons are updated within the same routine. Elapsed time of the last simulation run is calculated by querying the system function times(), which returns the current number of clock cycles. This value is subtracted from the value obtained at the last entry through the loop and is then divided by the clock rate. The simulator time function is initialized within the main routine's restart loop, as seen in Figure 6.3, so a start value can be obtained. The actual time calculations are shown in Figure 6.5. Once the time in seconds has been calculated, it is multiplied by the vessel's or weapon's speed and again by the sine and cosine of the course to determine a distance in the X and Z plane. This distance

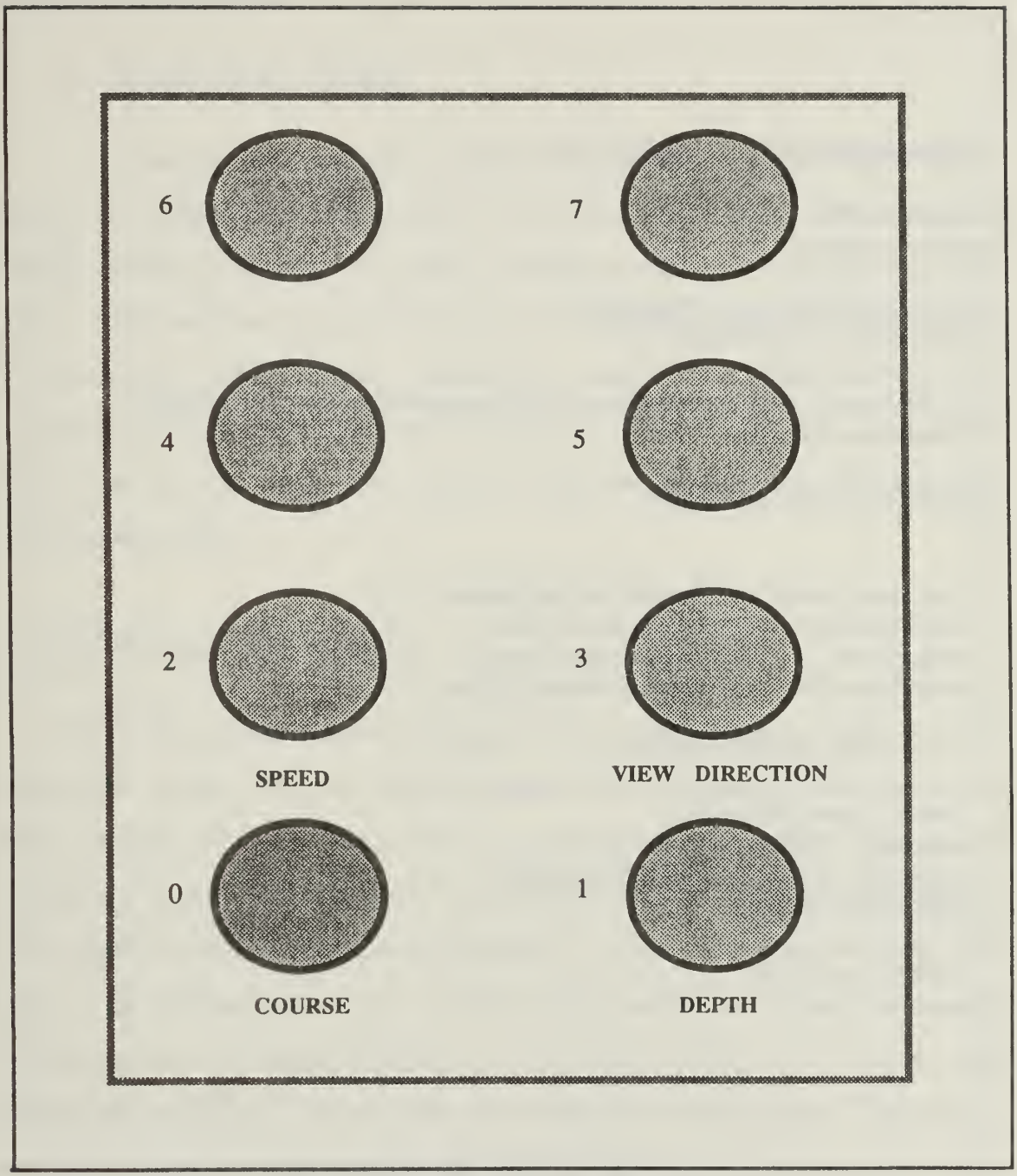


Figure 6.4 IRIS Dial Box

```
/* Global variables for time keeping */

long start_time;
struct tms timeinfo;

/* Process the loop time difference */
/* Called by updatepositions() */

/* Calculates the time expended during the last simulation loop and returns */
/* the elapsed seconds.                                                     */

float process_time_difference()

{
  struct tms timeinfo;  /* System time information     */
  float elapsedsec;     /* Returned time value         */
  long lastsec;         /* End time for simulation run */
  long elapsedhz;       /* Elapsed machine cycles      */

  /* start_time, global start time                      */

  lastsec = times(&timeinfo);
  elapsedhz = lastsec - start_time;
  elapsedsec = (float)(elapsedhz)/(float)HZ;
  start_time = lastsec;

  return(elapsedsec);
}
```

Figure 6.5 Process Time Calculation

is added onto the current gridX and gridY fields in the vessel or weapon packet currently being updated.

1. Stealth Mode Update

Selecting the stealth mode option from the main run menu automatically keeps the submarine five meters above the bottom. It is useful when navigating shallow channels, allowing the pilot to concentrate only on the course and speed dials. While stealthing the submarine aids in navigating the vessel, it does not keep the vessel from going aground. Grounding a vessel in the stealth mode has the secondary affect of causing the user to either retire from the simulator or restarting with a new operation area, since refloating operations in the program are performed by the stealth routine.

E. NETWORK UPDATES

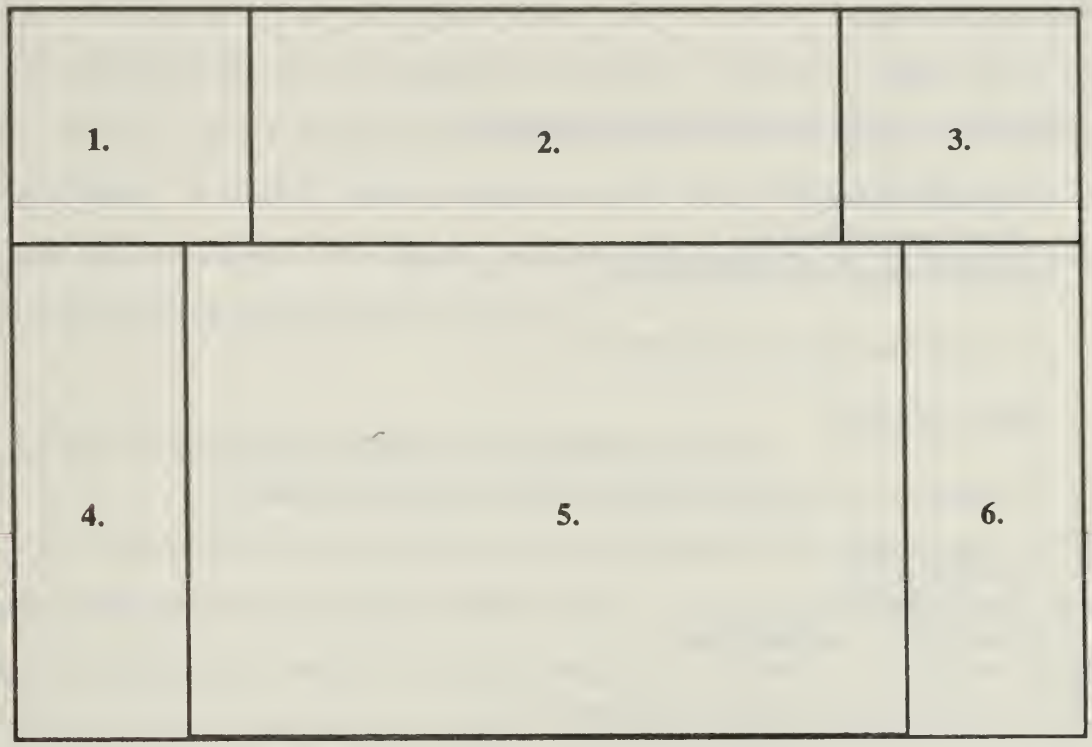
Networking, if selected, is processed after the vessel update has finished, as per Figure 6.2. Updates from the network include vessel initialization, course, speed, and depth changes, weapons firing and network maintenance messages. The current networking scheme utilizes a UNIX non-blocking broadcast socket. Once energized, the broadcast socket is kept until the program is exited. The "Turn Network OFF" flag on the run time menu sends a network KILL message and unloads the network vessel, wreck and weapon databases. The broadcast socket is not removed since UNIX does not allow the socket to be re-energized later in the program. Returning to the 2D placement operations, via the "restart with a new operation area" menu selection, sends a KILL message to the network. This is necessary since the timing for the 3D operations is stopped when the program returns to the 2D operations.

Updates for the driven vessel's course, speed and depth are sent upon completion of clearing the program event queue. Networking, as implemented, is discussed in the chapter 7.

F. FIVE NAUTICAL MILE PILOTING GRID

Once the network update has been completed, all six of the run time simulation displays are updated as described in the following sections. Figure 6.6 shows the relative positions of the six windows. The five nautical mile, 10,000 yard, piloting grid window is drawn first as shown in Figure 6.7. This window portrays the two dimensional data, color coded for height and depth, as seen from the vessel centered in the square. The vessel's current course is displayed as a red line of 3,000 yards length. The length is the same as seen along the depth window discussed later. The view angle is portrayed as a yellow V whose angle measure is equivalent to the current view angle selected by the user. This V is centered on the view direction. Rather than moving the vessel across the terrain, the terrain is scrolled beneath the vessel giving the user a constant 5,000 yard outward piloting grid.

The piloting grid is drawn using three hundred yard squares starting at a position 5,000 yards south and 5,000 yards west of the vessel. This centers the vessel on the grid. Three hundred yard grid squares are used to reduce the drawing time of the grid. A map grid drawn at one hundred yard resolution requires 10,000 polygons while the three hundred yard resolution requires only 1,089 polygons. While small land masses may appear and disappear as the driven vessel's position changes, the overall picture is quite navigable in conjunction with the other windows.



3D Operation Windows

1. Five Nautical Mile Piloting Grid
2. Visual Depth Finder
3. Sixty Nautical Mile Database Grid
4. Weapons Panel
5. 3D View Window
6. Control Panel

Figure 6.6 3D Operation Windows

```
redraw3Dscreens()

{
  /* Update the Pilot grid every other redraw */

  if (updatePilotMap)
    drawContourView();
  updatePilotMap = !updatePilotMap;

  /* Update Sixty nautical mile grid  */

   update_Sea2D();

  /* Update the Visual Depth Finder window every other redraw */

  if (updateDepth)
    drawSideView();
  updateDepth = !updateDepth;

  /* Update the maneuver 3-D window */

  if (DrawMesh)
    drawMeshManeuver();
  else
    drawManeuver();

  /* Display the polygon total */

  writepolys();

  /* Display the Frames per Second Meter Value */

  writeFramesPerSecond();
}
```

Figure 6.7 3D Window Update Schedule

Color coding is done according to the same color scale as was used to display the 2D sixty nautical mile database grid. A compromise was made by not storing the color at each point within the height array. This would have increased run time memory requirements by 1.44 megabytes assuming a character based legend was used. Instead, a binary tree coded if structure, Figure 6.8, was used for determining the grid color. A check is made to insure the point being colored is higher than the deepest water threshold. The piloting grid is initially cleared to the deepest water color and data points at this level are not drawn.

G. SIXTY NAUTICAL MILE DATABASE GRID

The next window to be updated is the sixty nautical mile database grid. This window portrays the user selected DMA database. The drawing time for this color height and depth coded window is approximately seven seconds. The window is used to show the user where his five nautical mile piloting grid is located on the sixty nautical mile grid. Figure 6.9 shows how the piloting grid location is tracked with a red outline square drawn in the overlay plane to show the piloting grid's location. The overlay plane can be cleared and redrawn without effecting the time consuming sixty nautical mile database grid. Current piloting area tracking can then be done without expensive, time consuming redraws of the database grid.

H. VISUAL DEPTH FINDER

The visual depth finder window is drawn next. This window displays graphically the terrain 3,000 yards ahead to 1,000 yards astern of the vessel directly on the vessel's dead reckoning line. The window is color coded to represent the water

```
/* The algorithm uses a binary tree to determine the color desired. */

    if (terrain[i][j] > -301)
      {
        if (terrain[i][j] > -1)
          if (terrain[i][j] > 130)
            if (terrain[i][j] > 195)
              if (terrain[i][j] > 250)
                  RGBcolor(BROWN);
              else
                  RGBcolor(DARK_GREEN);
            else
                RGBcolor(NEW_GREEN);
          else
              if (terrain[i][j] > 50)
                  RGBcolor(LIGHT_GREEN);
              else
                  RGBcolor(TAN);
        else
            if (terrain[i][j] > -151)
              if (terrain[i][j] > -21)
                  RGBcolor(LIGHT_BLUE);
              else
                  RGBcolor(BLUE1);
            else
              if (terrain[i][j] > -226)
                  RGBcolor(BLUE2);
              else
                  RGBcolor(BLUE3);

        /* color fill by rectangular system fill */

        rectfi(i,j,i+3,j+3);
        poly_count(1);
```

Figure 6.8 Binary Tree Color Coding

```
/* Update the oparea square on the 2-D overview map */
/* Called by redraw3Dscreens() */

/* This function will track the operation area on the 2D map */

update_Sea2D()

{
    /* Set the Window to the 60 Nautical Mile Grid Window */

    setwindow(MAP2D);

    /* Go to overlay mode and clear the old grid square */

    drawmode(OVERDRAW);

    color(CLEAROVERDRAW);
    clear();

    /* Redraw the Piloting Grid Location */

    linewidth(2);
    color(REDOVERDRAW);
    rect(my_sub.gridX - 50, my_sub.gridY - 50, my_sub.gridX + 50,
         my_sub.gridY + 50);

    /* Return to Normal Drawing mode */

    drawmode(NORMALDRAW);

     poly_count(1);
}
```

Figure 6.9 Sixty Nautical Mile Grid Update Routine

depths seen on both the sixty nautical mile database grid and the five nautical mile piloting grid. This window is used to help prevent the vessel from running aground. Note, if the user opts for the stealth mode, the submarine drawn in the depth window becomes an outline figure. This gives the user a visual cue of the vessel's stealth mode status.

The depth finder window is drawn as a fixed window of 4,000 yards length. Terrain heights from 1,000 yards astern to 3,000 yards directly ahead on the course line are drawn as brown quadrilaterals. The height of each terrain quadrilateral is determined by the difference between the -308 meter bottom depth and the terrain height at that point along the 4,000 yard line. Calculations for the course line traversed in this window are done with the stored course sine and cosine values. The vessel position is moved astern 1,000 yards by multiplying this value by the sine and cosine. Once the initial point has been determined, forty increments are done on the line resulting in the forty polygons shown in the window.

I. 3D VIEW WINDOW OPERATIONS

The next window to be updated is the 3D view window. The terrain is drawn in one of two user selectable methods. Mesh drawing uses the IRIS triangular mesh routines giving the most realistic view available from the simulator as seen in Figure 6.10. The terrain is shaded only according to the terrain model's light source reflection. While giving a very realistic view, it is sometimes hard to pilot on flat terrain. The grid square method, Figure 6.11, is supplied for such an occasion. It draws all the terrain as squares of two shades giving a checkerboard appearance. The terrain is still light shaded but traversal of flat terrain in the grid square terrain

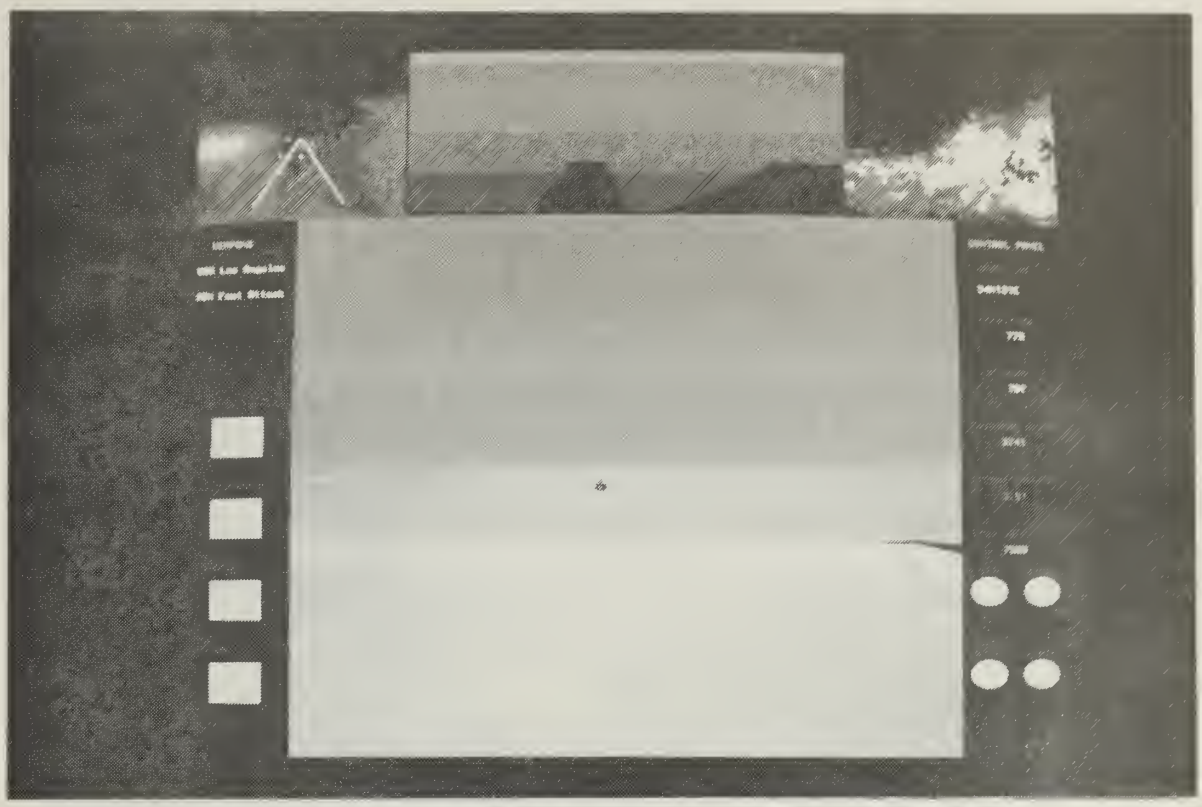


Figure 6.10 Triangular Mesh Terrain Drawing Method

Figure 6.11 Grid Square Terrain Drawing Method

gives obvious visual clues of vessel motion, often not seen with the mesh terrain. The user can freely toggle between the two methods by choosing the appropriate menu item.

Once the appropriate routine has been selected, terrain drawing begins. While much of the code for the drawing algorithms is shared, there is enough of a difference to discuss both versions. First, the grid square method. This method was the first completed and the mesh code is a modified version of this code made to draw the entire minor axis row with the mesh routines rather than just the individual squares.

1. Terrain Drawing Algorithm

The two terrain drawing algorithms used in the S&PV are based upon the same terrain drawing principles. The major differences are in the low level execution of the drawing algorithm. This section discusses the principles involved while the next two sections discuss the actual terrain drawing algorithms. Figure 6.12 is the reference figure for this section.

What we wanted to create is a terrain drawing algorithm that filled in only the area within the viewing angle in the quickest manner available. Our terrain algorithm applies resolution boundaries to each terrain sector drawn rather than to a particular quadrant as done in MPS [Ref 3]. This allows us to keep the number of terrain squares drawn within the viewing angle to the minimum required to draw the area of interest. The three major principles used in the algorithm are discussed below.

a. Five Degree Sectors

The two terrain drawing methods are first based on drawing the view angle as a series of consecutive, five degree terrain sectors. The sectors split the

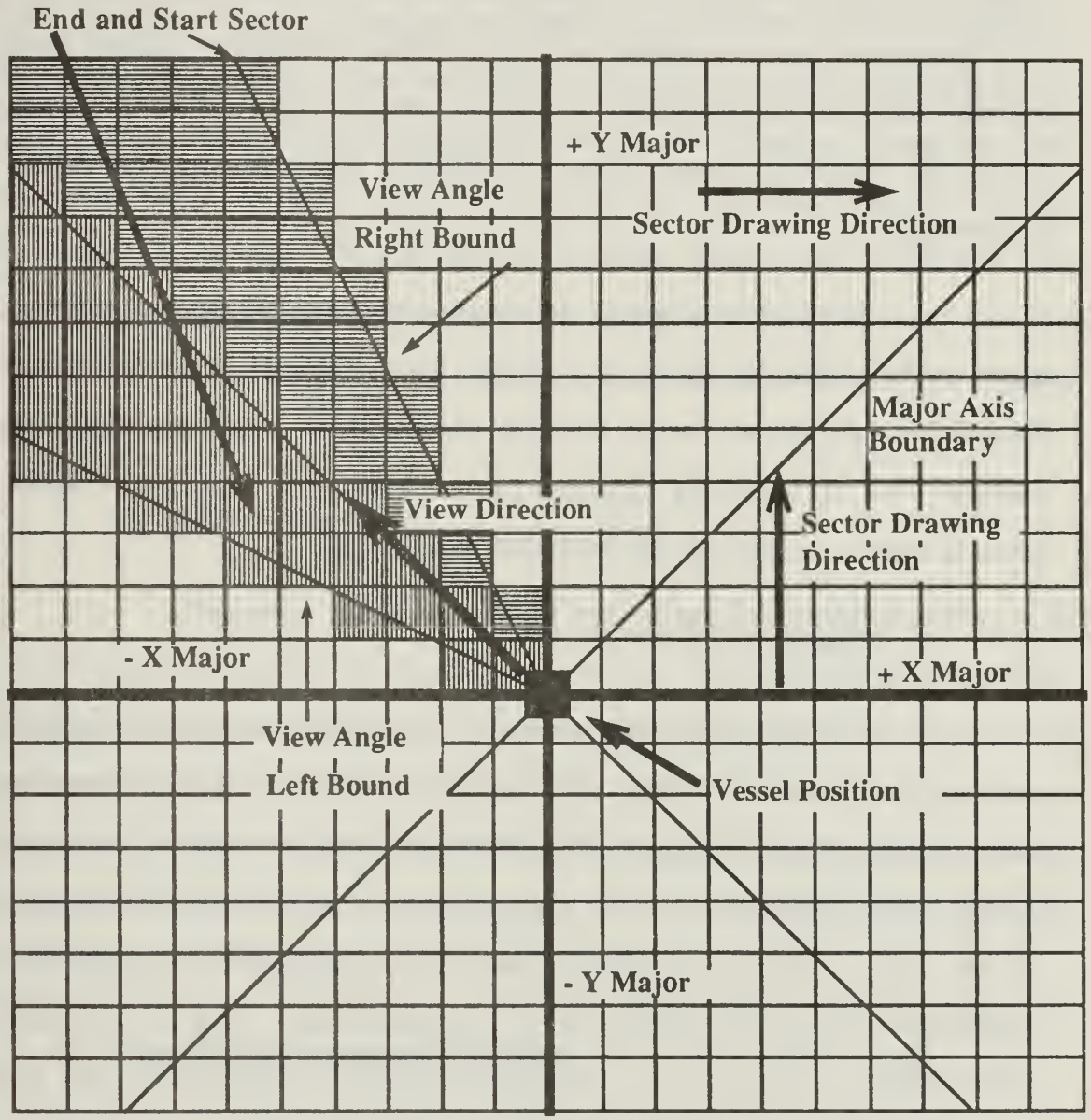


Figure 6.12 Sector Drawing Operations

viewing horizon into seventy two equally sized areas that are equivalent to a five degree arc on the viewing circle. The first sector, sector 0, covers from mathematical 0 to 5 degrees. The sector numbers advance counterclockwise up to sector 71, 355 to 0 degrees. In this way, a fifteen degree zoom angle is drawn as an area covering three sectors, the sector containing the view direction and the two adjacent sectors on either side of the view direction. The default viewing angle of forty five degrees is drawn with a nine sector area. The sector drawing information is stored within a lookup table.

Every sector drawn is drawn using resolution boundaries within each sector. This eliminates areas of expensive high resolution drawing where the distance from the observer would normally allow medium or low resolution terrain drawing. The MPS drawing algorithm contains such areas where high resolution terrain is drawn in medium and low resolution areas [Ref. 3].

b. Major and Minor Axis

The second principle inherent to the terrain drawing algorithm is the concept of major and minor axis. The major axis is defined in the two dimensional, mathematical sense as the positive X axis for angles in the range of 315 to 040, positive Y in the range of 045 to 130, negative X for 135 to 220 and finally negative Y for the range 225 to 310 as seen in Figure 6.12. The minor axis is merely the perpendicular axis to the major, again in two dimensions. This quadrant system buried within the sector system assures proper resolution boundaries are kept within the sectors. The sectors are drawn in a horizontal row for sectors in either the positive or negative Y major axis orientation and vertically for the positive or negative X major axis orientation.

c. Sector Drawing Lookup Table

The sector drawing lookup table contains the slope of the right hand vector of the selected sector. The slope of the left hand vector is the next sector's right hand vector slope. The slope is calculated as the increase or decrease along the minor axis of the sector for each unit increase along the major axis. This value is pre-calculated as either the sine over the cosine or cosine over the sine as appropriate for the major axis. The table repeats itself numerically every ninety degrees with the exception of the sign. Storing these calculations insures that the sectors join, reducing cumulative floating point errors and saving substantial calculation time that would have been necessary to calculate the sine and cosine at each increment along the sector boundaries.

d. Major Axis Splits

Originally, the viewing angle was drawn as individual sectors. To speed up the drawing process, the current version of both terrain drawing algorithms finds the starting and ending sector values and draws all the terrain between the two boundaries at once. The only exception to this rule is when the viewing angle splits a major axis boundary. Figure 6.12 shows a viewing angle drawn in two parts with the starting sector to the major axis boundary as one terrain drawing call and the major axis boundary to the ending sector as the second terrain drawing call.

2. Grid Square Drawing Method

Upon entering the grid square routine, function drawManeuver(), the 3D view window is set as the current output window as shown in Figure 6.13. The lighting model set up is initiated by switching the viewing mode from MSINGLE to

```
/* Set the 3D Viewing  window and turn on the lighting model */

setwindow(MANEUVER);

mmode(MVIEWING);
loadmatrix(unit_matrix);

lmbind(LIGHT0,Sun);

far_clipping_plane = (float)(YENTRIES * SCALE3D);

/* Set the appropriate perspective */
/* When making changes and you lose everything, the error is usually   */
/* here or in the lookat() command                                     */

switch(ViewAngle)
{
  case 15 :  /* 15 degree zoom angle */

      perspective(140,0.8,0.1,far_clipping_plane);
      NumberOfSectors = 1;
      Arc_Offset = 12.0;
      break;

  case 45 : /* Default 45 degree view angle. */
  default :

      perspective(450,0.8,0.1,far_clipping_plane);
      NumberOfSectors = 4;
      Arc_Offset = 4.0;
      break;

} /* end switch */
```

Figure 6.13 Lighting Mode and Perspective Command

MVIEWING and binding the sun light source. Note, that the lighting model is not turned on at this time. Next, the far clipping plane is established and the IRIS perspective command is issued, still in Figure 6.13. Actual parameters to the command are determined by the current view angle selected. The number of sectors to the beginning and ending sector, the field of view, and the drawing offset are determined by the view angle. The eye position, the lookat() command in Figure 6.14, is set according to the position and depth of the driven vessel. The view angle is always set by the viewing direction and not the current course. This allows three hundred and sixty degree horizon sweeps on any course, at any position. The start position of the drawing routine is set back from the actual vessel position a distance equal to the drawing offset, called "arc_offset" in Figure 6.13. This offset is necessary to remove the jagged edges produced at the edge of the view area.

a. Visible Quadrant and Water Surface Draw

After moving the drawing start position, the screen buffer and z-buffer are cleared as shown in Figure 6.14. The screen buffer is cleared either to sky blue or deep water blue depending on the height of eye of the vessel. Heights of eye above the surface level get sky blue and below, deep water. Routine drawSurface() is then called to accomplish two tasks. The first task it completes is determining the visible quadrant for later vessel and wreck placement. Vessels near the driven vessel but not in the calculated quadrant are not drawn. The second task it completes is drawing the water surface. With the lighting model still off, drawSurface() draws in the water surface grid lines. Since the lighting model shades all lines black, all grid lines had to be drawn with the lighting model turned off, allowing for the light blue colored lines drawn. Upon completion of the grid lines, the lighting model is turned on and the

```
/* The lookat is set for the current position of the sub in line with   */
/* the point one sin/cos away along the view direction.  All values */
/* are scaled according to sub.h description.  Currently no twist     */
/* angle is used.                                                      */

lookat(my_sub.gridX * SCALE3D,
      ydepth,
      -my_sub.gridY * SCALE3D,
      ((my_sub.gridX + my_sub.viewXinc) * SCALE3D),
      ydepth,
      -((my_sub.gridY + my_sub.viewYinc) *SCALE3D),
      0 );

/* Move the drawing start position back depending on the view angle */
/* to clear up the clipping on the left side of the 3-D picture          */

startX = my_sub.gridX - (Arc_Offset * my_sub.viewXinc);
startY = my_sub.gridY - (Arc_Offset * my_sub.viewYinc);

/* Turn on zbuffering for back surface removal and clear the Z-buffer        */
/* and the display screen to DARK BLUE if underwater else clear to SKY  */

if ( (my_sub.depth + my_sub.bridge) >= 0)
  RGBcolor(SKY);
else
  RGBcolor(DARK_BLUE);
clear();

zbuffer(TRUE);
zclear();

/* draw the ocean surface */

quadrant = drawsurface(mathView);
```

Figure 6.14 Eye Position, Arc Offset and Surface Call

actual water surface is drawn. The surface normal used to draw in the surface depends on the height of eye of the viewer. Above water, the surface normal points up, otherwise it points down. Only the surface area bounded by the designated quadrant is drawn. The surface of the water is drawn as one large polygon that is the size of the visible quadrant.

b. Terrain Sector Control Routine

After completion of the water surface draw, the terrain drawing begins with a call to drawTerrainSectors(). This routine calculates the start and end sectors as seen in Figure 6.15. It also calculates the length of the drawing run along the view's major axis. The second section of drawTerrainSectors(), Figure 6.16, determines which drawing quadrant the view direction and angle is in and passes the starting and ending sectors to the appropriate major axis drawing routine. The start position in the X and Z planes is passed from the drawManeuver() routine, where the offset was calculated.

The true view direction, used when piloting a vessel, is converted to mathematical degrees before the terrain drawing begins. This is necessary since true north is equivalent to 90 degrees on the mathematical scale and true degrees increase clockwise while mathematical angles increase in the counterclockwise direction. This conversion is done in the drawManeuver() routine.

c. Resolution Boundaries

Each terrain sector is drawn with three resolution boundaries as shown in Figure 6.17. In cases where the drawing distance is less than 7,500 yards, all three resolution areas are not drawn. The length of each resolution boundary is

```
drawTerrainSectors(view,startX,startY,NumberofSectors)

  float view,startX,startY;
  short NumberofSectors;

{
  short StartSector;   /* The "right" most sector */
  short EndSector;     /* The "left" most sector  */
  int run;             /* length of the run on the major axis */

  /* Calculate the start sector */

  StartSector = ((int)view / 5) - NumberofSectors;

  /* Check to insure the sectors range from 0 - 71 */

  if (StartSector < 0)
    StartSector = StartSector + 72;

  /* Calculate the end sector */

  EndSector = ((int)view / 5) + NumberofSectors;

  /* Check to insure the sectors range from 0 - 71 */

  if (EndSector > 71)
    EndSector = EndSector - 72;

  /* Calculate the run along the major axis that is required to get the */
  /* desired distance.                                                  */

  run = (int)(Arc_Offset + Draw_Distance);
```

Figure 6.15 Starting and Ending Sector Calculations

```
/* The math 45 to 130 degree region */
/* The major axis is the 2-D + Y axis(ie overhead view) */

if ((StartSector >= 9) && (StartSector <= 26))
  if (EndSector <= 26)
    plusYmajor(StartSector,EndSector,startX,startY,run);
  else
  {
    plusYmajor(StartSector,26,startX,startY,run);
    minusXmajor(27,EndSector,startX,startY,run);
  }

/* 135 to 220 degrees, -X is major axis */

else if((StartSector > 26) && (StartSector <= 44))
  if (EndSector <= 44)
    minusXmajor(StartSector,EndSector,startX,startY,run);
  else
  {
    minusXmajor(StartSector,44,startX,startY,run);
    minusYmajor(45,EndSector,startX,startY,run);
  }

/* 225 to 310 degrees, - Y is major axis */

else if ((StartSector > 44) && (StartSector <= 62))
  if (EndSector <= 62)
    minusYmajor(StartSector,EndSector,startX,startY,run);
  else
  {
    minusYmajor(StartSector,62,startX,startY,run);
    plusXmajor(63,EndSector,startX,startY,run);
  }
```

Figure 6.16 First Three Major Sector Drawing Control Routines

set by defined constants in the sub.h header file. High resolution draws all data points within the predetermined for loop bounds. This accounts for all of the one hundred yard square data points. Note, since four points are usually not planar, each terrain square is drawn as two triangles with the adjacent edge being the line from (i,j) to (i+1,j+1) in the two dimensional view point for high resolution. The width of the run between the left and right bounds of the sector at a specific point along the major axis is determined by use of the lookup table.

d. Medium and Low Resolution Drawing

The medium resolution area is drawn with two hundred yard squares, where the lower left hand corner is a multiple of two. This is done, again, to ensure the proper connection of the adjacent sectors and resolution boundaries. A skirt between the resolution boundaries is necessary to cover the difference in elevation from the odd center point drawn in the high resolution terrain and the line that connects the edge to the medium resolution as shown in Figure 6.17. This skirt has no width in the X and -Z planes since the hole that is created by the resolution boundaries is only along the vertical Y axis.

The low resolution terrain is drawn in the same manner as the medium resolution terrain. The start points for the low resolution terrain are all multiples of four to ensure proper connection to the lower resolution boundaries. A skirt between the resolution boundaries is also drawn.

e. Resolution Boundary Placement

One of the highest performance cost factors in the terrain algorithm is the decision where to place the resolution boundaries. While drawing everything as

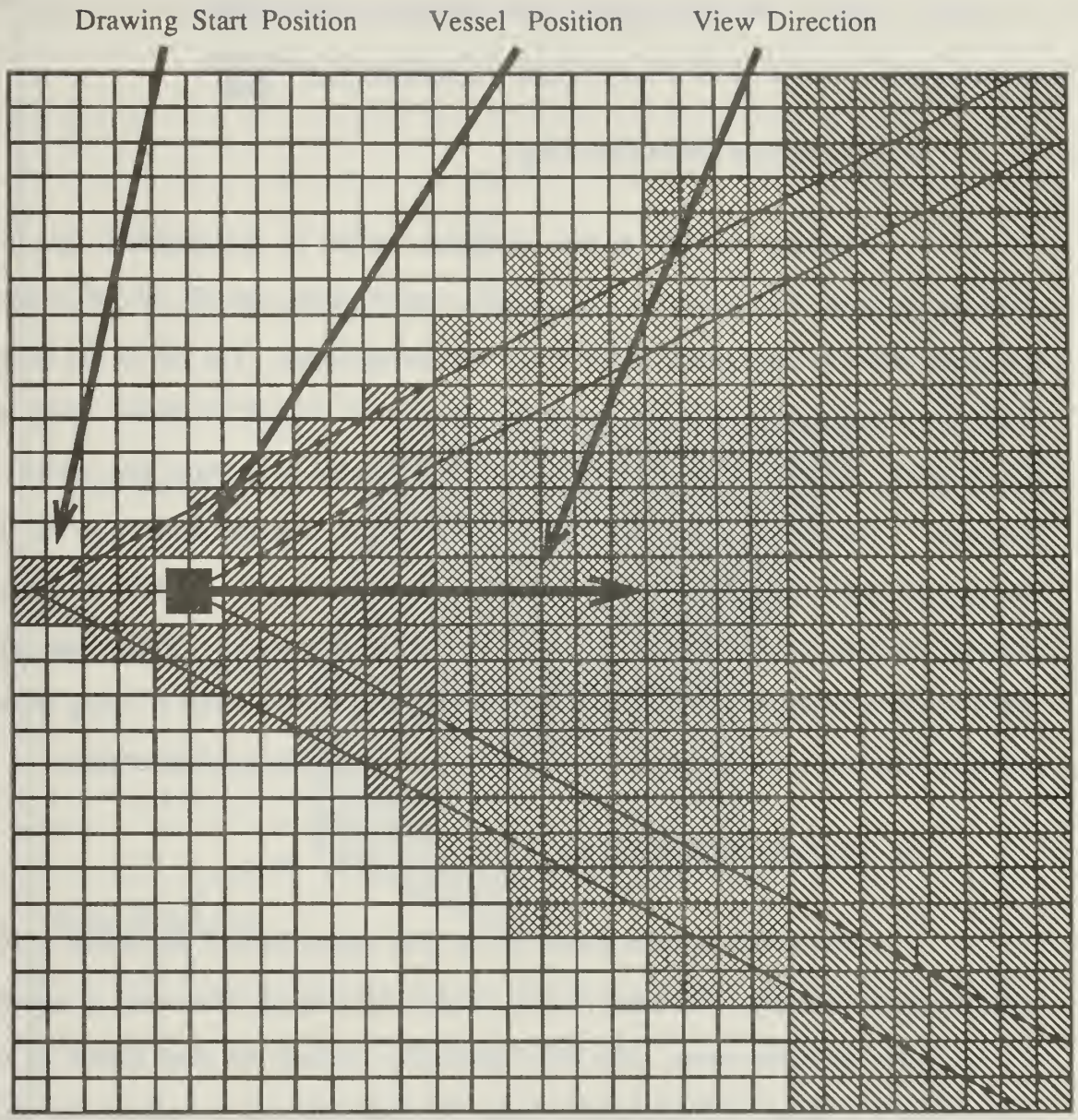


Figure 6.17 Resolution Boundaries and Drawing Offset

high resolution terrain may seem desirable, the computation time and numbers of polygons drawn is prohibitive. The current resolution bounds were set to give the best overall quality in the picture with the highest performance as possible. This is an obvious juggling act and changes to the resolution boundary lengths can easily be accomplished by changing the defined constants in the sub.h header file.

f. Resolution Skirt Drawing

Drawing of the terrain resolution skirts is accomplished by two different routines. One routine is used to draw vertical skirts, and the second to draw horizontal skirts as appropriate to the major axis being drawn. The job of the terrain resolution skirts is shown in Figure 6.18. Each skirt triangle is drawn using the bgnpolygon() routines since the triangular terrain patches only have one vertex in common. These vertical and horizontal skirt routines draw the skirts both for the medium to high resolution and the low to medium resolution. As with all terrain drawing routines, vertex normals are used for the lighting model. While this extra effort is not particularly noticeable in the grid square draw, the effect of using terrain normals in the mesh draw is considerable as discussed later in the mesh description.

h. Terrain Grid Squares

The terrain squares drawn for the grid square method are drawn as two adjacent mesh triangles. The same routine handles drawing all three sizes of squares for the three resolution areas. The IRIS bgntmesh() routines are used instead of the bgnpolygon() routines. The principle reason for using the mesh routines is speed. The square drawing routine fills the terrain squares in a checkerboard style. It alternates between the two defined terrain material definitions by reducing the lower

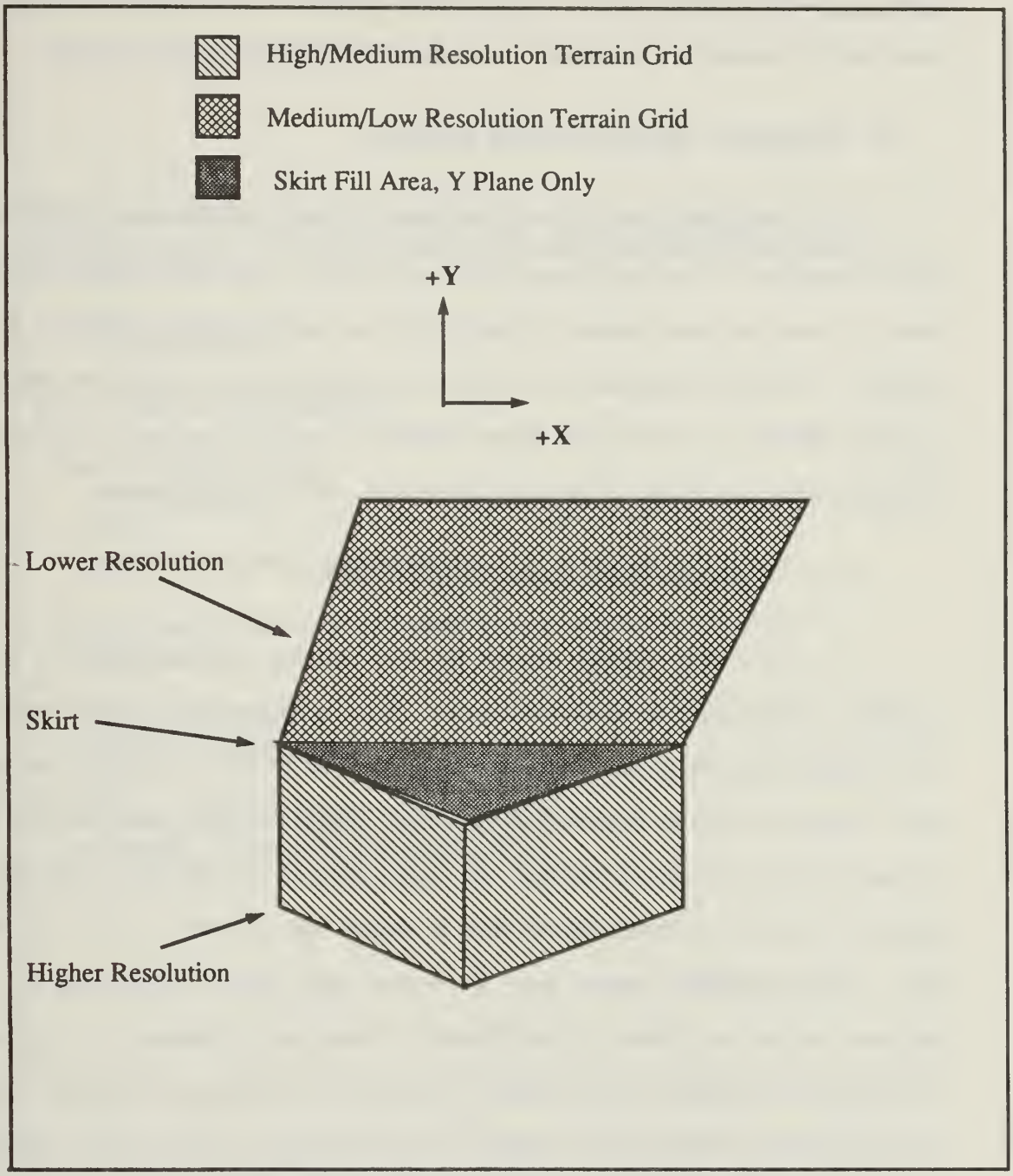


Figure 6.18 Skirt Drawing in the Vertical Plane

left hand corner to its high resolution terrain value and finding whether or not the square's indices are odd or even. Both triangles making up the square are shaded the same color by laboratory convention. A triangular checkerboard is quite possible.

3. Triangular Mesh Drawing Method

The triangular mesh drawing routines share all of the same upper level control structures as the grid square drawing method. No determination of which shade to color the mesh squares is made since we use the lighting model to do the shading. The vertex normals used within the drawing routine gives the terrain a smooth, realistic look. The grid square method is included because it can be very difficult to sense any motion in the three dimensional picture over flat terrain.

a. Mesh and Grid Square Differences

The first routines where the two drawing methods differ is in the quadrant drawing routines mentioned above. Since with mesh we want to draw the entire minor axis row at one time, the minor axis for-loops are placed in the mesh draw routines as seen in Figures 6.19 and 6.20. While the mesh draw uses the skirt routines from the grid square method, the square draw of the grid square method was converted into two routines, one to draw a vertical and the other a horizontal mesh row. The bgntmesh() routine uses the current input vertex along with the two previous vertices and creates a mesh triangle. Since the hypotenuse of the triangles within each mesh square must coincide between the vertical and horizontal meshes, the vertical mesh routines draws squares in the vertex order of right to left, bottom to top while the horizontal routine draws from top to bottom, left to right. If the vertex ordering does not coincide, such that the hypotenuses are not drawn in the same

```
/* High Resolution Run Section taken from the plusYmajor() routine in the      */
/* grid square method.  Note the twin for loops calling the drawNbyNsquare() */
/* routine.                                                                    */

  /* Hi Resolution Run */

  for (j = 0; j <= even_hi_res; j++)
  {
    for ( i = (int)(left - 0.5);
          i <= (int)right;
          i++ )
      drawNbyNsquare(1,i,(int)startY + j);

    /* Increment left and right according to there slopes with whole */
    /* increments.                                                   */

    if (EndSector == 26)
      left = left - lookup[EndSector + 1][SLOPE];
    else
      left = left + lookup[EndSector + 1][SLOPE];
    right = right + lookup[StartSector][SLOPE];
  }
```

Figure 6.19 Grid Square Drawing Loop

```
/* This code fragment performs the same control and drawing as in figure 6.16.    */
/* Note how the inner for loop of figure 6.17 has been replaced by the call to     */
/* drawHorzMesh(). This allows the entire minor axis draw to be accomplished */
/* in one call fully utilizing the triangular mesh routine's full abilities.       */


  /* Hi Resolution Run */

  for (j = 0; j <= even_hi_res; j++)
  {
    drawHorzMesh(left - 0.5,right,(int)startY + j,1);

    /* Increment left and right according to there slopes with whole */
    /* increments.                                                   */

    if (EndSector == 26)
      left = left - lookup[EndSector + 1][SLOPE];
    else
      left = left + lookup[EndSector + 1][SLOPE];
    right = right + lookup[StartSector][SLOPE];
  }
```

Figure 6.20 Mesh Drawing and Control Fragment

orientation, z-buffer conflicts creates severe flickering within overdrawn terrain squares where adjacent sectors overlap.

All resolution boundary operations are kept within the triangular mesh control routines since this function resides in the major axis computations in the plusXmajor() related routines. The increase in performance with the mesh routines is approximately one and a half to two frames per second with the lighting model.

4. Encountered Drawing Problems

The major problems that we have encountered with our data has related to the shear size of the databases. The height information for the sixty nautical mile grids requires 2.88 megabytes of storage, while the normals for these points require nineteen megabytes. Currently our IRIS 4D/70GT's have only eight megabytes of system memory, with the exception of gravy4 which has sixteen megabytes, of which approximately half is used by the operating system. Major increases in speed should be attained with additional memory. Higher processor speeds should also help in the long run.

5. 3D Vessel Drawing

After the terrain drawing has been completed, all tracked vessels and wrecks within the calculated visible quadrant are drawn. First all vessels in the contact list and network contact list are drawn into the five nautical mile piloting grid as green dots. Since the dots are drawn in front buffer mode and there is a time difference between initial completion of the overhead view and the time the vessels are added to the grid, the green vessel position indicators blink, making vessel

tracking easier. All wrecks are done the same way with the exception of the color, which is red.

Once the piloting grid has been updated, vessels, wrecks and weapons in the visible quadrant are drawn. Each of the above data packets has a unique defined constant for the type of vessel. This value causes the controlling switch statement to select the appropriate vessel drawing routine as seen in Figure 6.21. The current system stack is first pushed onto the system stack. The rotation for the platform's current course and the translation to its position and depth using IRIS rot() and translate() commands are accomplished next. Finally the vessel is drawn and the system stack is popped returning the original system matrix. The process is then repeated for each item in the visible quadrant.

6. Final 3D Window Operations

Upon completion of the vessel drawings, z-buffering is turned off as well as the lighting model. The light in the model has to also be turned off or else "restarting with a new operation area" causes an lmbind runtime error. The viewing mode is returned to MSINGLE until the 3D drawing is again started. MSINGLE mode is required for the system popup menus to work.

J. POLYGONS PER FRAME AND FRAMES PER SECOND

The last section of the screen updates places the current polygon count and frames per second calculation on the control panel window. These values are averaged and updated every ten times through the display loop. These values are used in algorithm evaluation and scene complexity calculations.

```
drawVesselProfiles(profile)

  short profile;   /* Profile from vessel and wreck packets */

{
  /* Switch to draw the vessel */

  switch(profile)
  {
    case LOS_ANGELES :  /* Draw a Los Angeles Class sub */
          draw_LosAngeles();
          break;

    case TYPHOON :        /* Draw a Typhoon Class Russian Sub */
          draw_Typhoon();
          break;

    case OHIO :           /* Draw a Ohio Class Sub */
          draw_boomer();
          break;

    case SPRUANCE :       /* Draw a Spruance Class Destroyer */
          draw_spruance();
          break;

    case KITTY_HAWK :   /* Draw a Kitty Hawk class carrier */
          draw_kitty_hawk();
          break;

    default:
         break;

  } /* end switch */
}
```

Figure 6.21 Vessel Draw Switch Statement

VII. NETWORKING OPERATIONS

Networking within the S&PV is required for multiple graphics simulators to be operating simultaneously. Networked workstations share their vessel contact, wreck and weapons information and allow for target acquisition and persecution for vessels created on any of the workstations.

A. OVERVIEW

Networking is accomplished over Ethernet with a non-blocking Unix socket with the associated message overhead. The socket is checked when the networking flag is energized. Each message received from another host is processed according to the type of message it is. Each message type has its own length, and therefore a reception routine for each kind of message. All of the network message, send and receive, operations are in the file netMessages.c. The network checking routine is in checkNetwork.c while the low level, socket specific, routines are in the file networking.c.

The low level networking code used to initialize, send and receive messages and kill the socket was initially developed by Captain Randy Strong, USA [Strong, 1989]. His code was modified to work only between the graphics workstations in the Graphics and Video Laboratory. A check to prevent processing of a workstation's own messages is provided within the low level routines. This is necessary since Ethernet echoes the messages sent to the sender.

```
while (recv_broadcast_message(packet))
{
  switch (packet[0])
  {
    case HELLO :          /* Hello Message */
       receiveHello(packet);
       break;
    case HI :             /* Hi message     */
       receiveHi(packet);
       break;
    case VSL_INIT :       /* Vessel Initialization Message */
       receiveVslInit(packet);
       break;
    case WRK_INIT :       /* Wreck Initialization Message */
       receiveWrkInit(packet);
       break;
    case VSL_UPDATE :  /* Vessel Update Message */
       receiveVslUpdate(packet);
       break;
    case RESEND :         /*  Resend Vessel message */
       receiveResend(packet);
       break;
    case HOSTKILL :       /* Host Going Off the Line Message */
       receiveKill(packet);
       break;
    case WEAP_INIT :      /* Weapons Initialization Message */
       receiveWeapInit(packet);
       break;
    case MOVE_MYSUB : /* Move the Driven Vessel to the Wreck List */
       receiveMoveMySubToWreck(packet);
       break;
  } /* End switch. */
} /* End of while recv_broadcast_meassage() statement. */
```

Figure 7.1 Check Network Switch Routine

The upper level routines handle the creation and decoding of the network messages. As each message is received, the routine checkNetwork(), Figure 7.1, selects the proper decoding routine specific to the type of received message. Each message type described below comes in two types, the sending routine and its receiving routine. Actual sending and receiving of messages is done in the low level routines mentioned above.

B. HELLO MESSAGE

The "Hello" message notifies the network that a new workstation is coming on line. The host name of the new workstation is sent as part of the message. Upon reception of a "Hello" message, each workstation already on the net sends out a "Hi" message followed by vessel, wreck and weapons initialization messages.

C. HI MESSAGE

The "Hi" message is sent out from workstations already on the net to the new workstation just entering. This message tells the new station who is on the net. Reception of a "Hi" message causes the new workstation to send out its vessel, wreck and weapons list to the net.

D. VESSEL INITIALIZATION MESSAGE

The vessel initialization message sends all of the necessary information about a vessel for the vessel to be recreated by the other workstations on the net. Vessels are uniquely identified with two key fields. The first is their individual vessel packet identification number that was assigned by the host workstation at the time of the

creation of the vessel. The second key field is the name of the host that created the vessel. This allows all workstations to start with the same identification number series which removes long initialization processes where machines on the net must decide where their packet identification numbers must start.

Reception of a vessel initialization message is processed in one of two possible ways. First, if the vessel is new to the receiving workstation, a new network vessel packet is created and filled with the appropriate data sent to the machine. Calculations of the sine and cosine of the network vessels course are done during the data filling operations. If the vessel already exists, the vessel's position, depth, course and speed are updated according to the new message's information.

E. WRECK INITIALIZATION MESSAGE

Network wreck initialization is done in the same way as vessel initialization is done. The major difference in the message is the lack of a speed field which is obviously zero knots for a wreck. This message along with the move driven vessel to wreck list message is primarily used to insure consistent displays on all network machines.

F. WEAPONS INITIALIZATION MESSAGE

Weapons initialization messages are similar to the vessel initialization message with the exception of two added fields. These fields are the elapsed time on the weapons run and the maximum run time for the weapon. These fields are necessary since all of the weapons shot in the CCWF have a fixed time frame they can be active.

G. DRIVEN VESSEL UPDATE MESSAGE

This message provides to the network the changing operating parameters of the vessel being driven on the present host. The updates sent to the net are vessel course, speed and depth. Reception of an update message is first switched by the message type, then by the update type. Then the appropriate changes to the network vessel packet representing the driven vessel are made.

H. MOVE DRIVEN VESSEL TO WRECK LIST MESSAGE

This message is necessary to keep the network informed of changes to a host machine's driven vessel in ground collision conditions. While collision detection is done on each machine, changes to the operating parameters of a driven vessel causing a sudden grounding may be transparent to the network. This message formally declares a collision to the net, creating a wreck at that position. It then renumbers the driven vessel's packet identification number and a new vessel initialization message is sent to the net. Note, this assumes that the user opts for either the "restart with a new oparea" or "refloat the submarine" options on the grounding menu. Retiring from the simulator does not create the message as a network kill message is sent instead.

I. RESEND MESSAGE

A resend message is sent if upon reception of a vessel update message, the vessel being updated is not in the network vessel database. This message contains the packet identification number and the name of the host computer. Reception of the

resend message causes the host computer of the vessel to resend a vessel initialization packet, usually being the host's driven vessel.

J. KILL MESSAGE

The kill message is sent when a host has decided to leave the networking mode. This occurs as the result of one of three cases. They are user selection of the "turn off network" main menu option, the simulator is moving move back into the 2D operations mode, or the program is being exited.

Reception of a kill message causes the receiving host to search its network databases and delete those vessels, wrecks and weapons created by the host going off line. The kill message does not shut down the socket. A socket can only be opened once each program run. Only during actual exit operations is the network socket released.

VIII. THREE DIMENSIONAL VESSEL ICONS

One of the basic values of a simulator is the ability to maneuver vehicles and equipment without the physical cost factors involved. Training can be accomplished without the risk of injury or equipment damage. As a training aid, it is necessary to make a simulator as realistic as possible. Realism increases the training value of the simulator and makes it a more useful tool for the commander. No simulator will ever replace real experience but a simulator can better prepare individuals for a wide range of expected situations. Equipment modeling is essential to good simulations. The commander who can see the situation as it develops has a marked advantage over his opponent. Ideally, the commander would like to train with and against the same equipment he would use in a real situation, so the need for a library of enemy and friendly equipment is essential.

The S&PV system provides the user with several realistic 3D vessel icons which add realism to the program. The vessels currently available to the user are the Spruance class destroyer, the Kitty Hawk class aircraft carrier, the Ohio class ballistic missile submarine, the Los Angeles class fast attack submarine, and the Soviet Union's Typhoon class ballistic missile submarine. These are just a few of the many vessels needed for a complete simulation library. The current vessels are adequate for providing the user the ability to maneuver vessels and see what other vessels look like in relation to the three dimensional terrain. More 3D icons of vessels and weapons are needed.

There are several things to consider when modeling vessels and weapons. A primary consideration is the number and size of the polygons required for each object. Objects that are complex need a greater number of polygons. A greater number of polygons increases the time it takes the processor and graphics pipeline to draw the object. Depending on the speed of the graphics machine and the complexity and number of objects, the simulation may run too slow to be of any real value.

Another consideration in modeling is the degree of accuracy of the drawing. While similar to the number of polygons problem, it is a different problem. For example, there are several different classes of American submarines and their basic hull shapes and sizes are similar. A 3D icon needs to bring out these differences, yet not be polygon intensive. The resolution needed in a given situation should also be considered. Vessels at greater distances from the viewer cannot be seen as well regardless of the detail of the vessel so there is no need for high detail polygons to be drawn at great distances. As the vessel moves closer, the more accurate drawings can be used. Such resolution and vessel dependencies save drawing time at the farther distances but require that the programmer have resolution boundaries in his program.

A final consideration is drawing and storing these models in an efficient manner. Graphics programs in general are memory and processor intensive. This necessitates the programmer be conscious of efficiency in coding and storage. As computer graphic capabilities improve, these considerations will have less impact. However, to take full advantage of the current graphics systems, these considerations must be implemented.

There are currently four initiatives at the Naval Postgraduate School to overcome some of the above considerations. The first initiative is the development of a standard file format to store and draw objects. The second is using a 3D digitizer camera to take pictures of objects and then use the collected data to draw the object. The third initiative is an attempt to build a 3D icon construction system to put together objects and create vessel icons. The fourth initiative is the graph paper technique of drawing the object and plotting the points.

A. STANDARD FILE FORMAT

There are several advantages of using a standard file format for objects rather than continuously using the drawing code. A standard draw polygon routine for three polygons using the Silicon Graphics graphics library is found in Figure 8.1. The code draws three adjacent polygons and includes the polygon surface normals needed for the lighting model. This code is simple and repetitious. Also notice that arrays are needed to hold the normal and vertex values for each polygon. These arrays are accessed during the draw polygon routines. This is not a bad way to handle drawing routines, but using a standard format is better. Figure 8.2 shows the same three polygons stored in a standard file format. The standard file format can be read in by one, fixed routine and displayed with another fixed routine. Having only one drawing routine allows for easier object modification and repetition of code is eliminated. This format allows the user flexibility in manipulating point data with its free formatted data allowing for easy adaptation to other systems.

The standard file format reduces the amount of stored code and the size of the data structure. The only real disadvantage of the standard file format, is dealing with

```
static float ship_hull[5][3] = {   0.0,    0.0,     0.0,
                                  -0.5,    0.20,   -0.11,
                                  -0.5,    0.11,   -0.22,
                                  -0.5,   -0.11,   -0.22,
                                  -0.5,   -0.11,   -0.11  };

static float ship_polygon_normals[3][3] = {     -0.727607,   0.485071,  -0.485071,
                                                -0.707107,   0.000000,  -0.707107,
                                                -0.557086,  -0.742781,  -0.371391   };

draw_ship()
{
  bgnpolygon();
      n3f(&ship_polygon_normals[0][0]);
      lmbind(MATERIAL, grayTwo);
      v3f(&ship_hull[0][0]);
      v3f(&ship_hull[1][0]);
      v3f(&ship_hull[2][0]);
  endpolygon();

  bgnpolygon();
      n3f(&ship_polygon_normals[1][0]);
      v3f(&ship_hull[0][0]);
      v3f(&ship_hull[2][0]);
      v3f(&ship_hull[3][0]);
  endpolygon();

  bgnpolygon();
      n3f(&ship_polygon_normals[2][0]);
      v3f(&ship_hull[0][0]);
      v3f(&ship_hull[3][0]);
      v3f(&ship_hull[4][0]);
  endpolygon();
}
```

Figure 8.1 Polygon Draw Routine

```
/* The order of the Standard Format is: polygon followed by the polygon normal,*/
/*  the number of vertices of the polygon,  and the position of each vertex.      */
/*   Each vertex is on a separate line in x, y, z order.                          */


polygon
  -0.727607   0.485071  -0.485071
  3
    0.000000    0.000000    0.000000
  -0.500000  -0.220000  -0.110000
  -0.500000    0.110000  -0.220000

polygon
  -0.707107   0.000000  -0.707107
  3
    0.000000    0.000000    0.000000
  -0.500000    0.110000   -0.220000
    0.500000  -0.110000   -0.220000

polygon
  -0.557086   -0.742781  -0.371392
  3
    0.000000    0.000000    0.000000
  -0.500000  -0.110000  -0.220000
  -0.500000  -0.110000  -0.110000
```

Figure 8.2 Polygon Standard Format

drawing errors. With the current standard file format every misplaced point must be corrected at each polygon it occurs in. In the coded format, only the point array value would have to be changed. To get around this problem, all corrections were made to the IRIS polygon code then converted to the standard format using a locally written routine.

Currently, the file format shown exists without the ability to display vessels in the format within current simulators. Work along this line has progressed to the point where objects can be drawn and manipulated in three dimensions with lighting using a standard format. This is useful for examining an object before using the object in a simulation program. This work will be useful when objects can be built on screen, then stored in a standard format and the drawing routine that uses the standard format is incorporated into the simulators. Clearly the standard format is in our future efforts.

B. DIGITIZED IMAGE

There is also an effort at the Naval Postgraduate School to take 3D digitized pictures of various ship models and convert the digital data into a standard format for drawing. This is a complicated process because the digitizing produces thousands of polygons for each object. The objects currently employed in the submarine simulator range from 75 to 136 polygons and represent accurate models. The digitized images would be very accurate and realistic, but drawing time would grind the simulation to a halt. It is our experience that limiting the polygons for each object to a maximum of 150 polygons gives the necessary detail and does not significantly effect simulator speed. The polygon count needs to be limited since several objects could be seen

within the same screen, significantly slowing down the simulation. A realistic goal is to limit the number of total 3D icon polygons to one thousand. This number of polygons enables the user to see seven to ten objects and maintain smooth motion. Limiting these polygons is complicated but necessary and a method to do this with 3D digitized models will be useful.

C. LEGO® BUILDING SYSTEM

Another way to produce three dimensional objects is to construct the 3D icon on the screen. It is possible to customize the image and then capture the image into a standard file format. This is the basic concept of another project at the Naval Postgraduate School.

Many of the vessels currently in the Navy's inventory have common attributes. Each class of ship is configured to satisfy a certain mission, with certain major components of the ships common. The ship building system under current development will contain a library of common ship parts that can be selected and connected graphically, like LEGO® blocks. For example, several ship classes have common hulls with different superstructures, so only the superstructures need to change. The building system will give a choice of what superstructures are available to construct the ship. The same goes for weapons systems and antenna structures. Other capabilities of the building system will be the ability to scale models to fit the coordinate system of different simulators, the ability to view the model with a lighting model to ensure polygon normals are situated properly, and the ability to construct new parts of ships as new equipment is developed. This is an important project because it will save time in icon construction and reduce repetition of effort.

D. GRAPH PLOTTING

If the modeling process can be accomplished with the aid of a computer there is less of a chance of an error and a good chance of saving time. Since the above techniques are in there embryonic stages, the modeling for the submarine simulator was accomplished with two dimensional drawings on graph paper. This technique is time consuming and lends itself to human error. Each point on a model requires a value for the x, y, and z axis. It is difficult to derive these values from a single two dimensional drawing. Three views of each model were required to develop adequate detail and realism. The drawings and photos for each model in the submarine simulator are provided in Appendix A. The points for each model are annotated on the drawing and the values of each point are provided on the figures following the drawings. There are several concepts to keep in mind when drawing models, the key concepts are modularity and efficient use of polygons to maintain accuracy.

Modularity is a key concept in any program and in modeling is especially important. When drawing objects, always take one section at a time before moving on to another section. The same idea is necessary with three dimensional programming. It is much easier to draw sections of an object separately then translate the objects together to form a whole object. This makes the object much easier to modify if there is a change in specifications or if an error was made. Simplicity is the key. If an object can be broken down into components and each component drawn, then there is less chance of an error.

Efficient use of polygons with relation to accuracy is another important consideration when building models. It is very important to make the vessel look realistic but not to the point of grinding the simulator to a halt. It is not necessary

that every detail be considered unless the detail is a distinguishing feature. When drawing the Ohio Class submarine and the Los Angeles class submarine, the same hull was used. There are two big structural differences between the Los Angeles and the Ohio classes. The first being the length, 560 feet for the Ohio class and 360 feet for the Los Angeles class, and the second being the long flat top of the Ohio class where the missiles are housed. The length in this case was not the important distinguishing factor. The vessels would identical unless positioned together so the relative lengths can be determined. The Ohio was modified slightly to emphasize the long flat top and then colored differently from the Los Angeles to bring out the difference. These changes cost six polygons which is not significant. All of the other sections of the submarines are identical except scaled to match the size of the Los Angeles. In this case accuracy and polygon count were not compromised.

One of the difficult problems faced in drawing models is the accurate depiction of curved surfaces. A large concentration of polygons are situated on the bow of the submarine because it is difficult to simulate smooth curved surfaces without using many polygons. Small triangles were used to insure the polygons were planar and to give a smooth look to the bow. Half of the polygons on the bow could have been eliminated if planar rectangles were used. This extra cost could not be avoided for realism. Rectangles did not look as good as the triangles. In contrast, the remaining sections of the hull of the submarine are done with rectangles and form the octagonal tube of the hull. Unlike the bow, the octagonal tube is quite adequate for the simulation. There are no three dimensional sphere or tube routines in the graphics library, therefore these shapes are constructed using three dimensional lines and polygons. This is an example where the polygon count had to be increased to render an accurate drawing.

The Kitty Hawk class aircraft carrier is an example of how few polygons are needed when eliminating unnecessary detail and where curved surfaces are not necessary. The 1000 foot Kitty Hawk uses only 76 polygons where the 360 foot Ohio class submarine uses 132 polygons. This is possible because the outstanding features of the Kitty Hawk are its size and its shape. The entire superstructure of the carrier uses only 15 polygons because it is not the important and distinguishing feature of the vessel. The Spruance class destroyer uses only 45 polygons for the hull and another 31 for the superstructure. Again this is because curved surfaces are not necessary for the Spruance vessel simulation. The Spruance does however have two large communication and sensing equipment towers which are drawn with the line drawing routines. Each tower is 18 polygons which sends the Spruance polygon count to 102 polygons. Unlike the Kitty Hawk, the superstructure of the Spruance is a distinguishing feature and increased detail to the superstructure is critical to the model. Added details like gun turrets and sonar domes are features which will eventually be added to the Spruance when the LEGO® building system is developed. The Spruance and the Kitty Hawk are examples of eliminating accuracy to lower the polygon count, yet still maintain a suitable object for simulation.

E. NORMALS AND LIGHTING

A dramatic improvement to any simulator is the incorporation of a lighting model. Without a lighting model, objects in a three dimensional simulator look like two dimensional cut outs of objects. A lighting model allows the object to be shaded relative to a common light source. This adds to the perception of depth. As an object moves away from the light source, the object gets darker relative to its closer position. All this adds to the realism of the simulation. In order to take advantage of

a lighting model, the normal for each polygon needs to be calculated. The normal is calculated by finding the unit vector perpendicular to a planar polygon. The normals are necessary because it is the angular difference between the light source and the polygon normal that determines the exact color for a particular polygon.

Currently, the polygon normals are included in the polygon drawing routines and the standard file format. This is accomplished by creating an array of normals for each polygon drawn in the order they are drawn. Using normals allows each polygon to actually have a different color with respect to the light source. A better way to use the polygon normals to produce a lighting effect is to calculate the vertex normals. To calculate the vertex normals the average of all the polygon normals associated with a particular vertex is taken. This must be done for each vertex. With vertex normals, individual polygons become less visible because they are blended with the adjacent polygons to give a smooth shaded appearance. This not only makes the lighting model more effective, but also smooths the rough edges of the object to make the ship models look more realistic. The use of vertex normals has not yet been incorporated into the three dimensional icons of the simulator.

Currently, modeling is very complicated and lends itself to artistic impression. With the efforts of creating a standard file format, a LEGO® building system, and a digitizing system, the accuracy and efficiency of object modeling can be improved. These systems will have to be able to balance accuracy with efficiency in order to make the results usable and not rely on advancements in technology to improve the speed of simulators. These improvements are necessary to increase the quality of current simulators adding realism and training value, and to take full advantage of advancing technology.

IX. PERFORMANCE

Since this is the preliminary work on the S&PV system, we used the development process to evaluate various means of data storage, terrain drawing methods and normal calculations. The highest rated versions are incorporated within the S&PV system.

In the test results that follow, the following program setups were used. The course and view angle were tied together. Stealth mode was not energized. The view angle was left at its default forty five degree view and the drawing distance was left at its default 7,500 yard range. The polygon count, averaged over ten draws, was between 1,800 and 2,000 polygons depending on the view angle. Test results given are for a 000 degrees true course with the polygon count specified. Networking mode was off. Grid square and mesh terrain drawing methods were toggled as indicated. No contact vessels were being drawn. Figure 6.1 gives a good feel for the complexity of the picture.

A. MEMORY REQUIREMENTS

First and foremost, the S&PV system is memory intensive. Its database of 1,201 by 1,201 sixteen bit integers requires 2.88 megabytes of memory storage. Since it is designed to not limit the user to any one location in the database, all of the terrain data is kept in memory. To effectively light the terrain, vertex normals are used for all the terrain points. Each normal is three 32 bit floating point normals, increasing the memory load to well over twenty one megabytes. Next, the program

with all of the graphics library routines included, must also be in memory. The S&PV is currently over 20,500 lines of C code and the executable file is over 700 kilobytes. Our system on gravy4, an IRIS 4D/70GT, has sixteen megabytes of system memory that must be shared with the operating system. While the IRIS is a multi-user, multi-tasking machine, any activity on gravy4 outside the S&PV has a noticeable performance degradation on the program. Virtual memory paging is the most probable cause.

B. 2D AND 3D OPERATIONS CHARACTERISTICS

1. 2D and Normal Calculation Operations

Test runs on the S&PV using the utility gr_osview have shown the S&PV to have two very different personalities. During surface normal calculations and 2D placement and selection processes, processor wait time is entirely composed of memory swap activity. While the surface normals array does not have to be in memory for 2D placement and selection operations, the entire height array is called upon on numerous occasions. Separate arrays for the normals and height data was found to be the fastest method available for all operations.

2. Terrain and Normal Data Storage Methods

Combining the terrain heights and terrain normals arrays into one array of records resulted in slow, memory intensive, halting displays in the 2D operations. 3D operations were also slower. Using two arrays has still caused an approximate ten second delay from the time the 2D overview map is drawn until the first program

menu is displayed for the user. After this delay, redrawing the 2D map does not cause any extra delay.

3. 3D Operations Characteristics

During the 3D operations, the processor wait time is nearly entirely composed of waiting for the graphics pipeline to unload enough graphics data to accept more data. Faster processor speeds, both numerically and graphically, would help the overall performance. Higher processing speeds will not reduce the need for more memory. The reduction of page swapping brought about by additional memory should help free up more processor time.

4. Shared Graphics and C Libraries

To reduce the memory requirements of the S&PV, we now compile the program using the shared C and graphics libraries. This has reduced the executable file by over 500 kilobytes. While the shared libraries should decrease performance, with their inherent indirection, our performance measurements noticeably increased by approximately one frame per second. This again points to the lack of memory for the system and the poor performance of UNIX when system memory is full.

C. TERRAIN DRAWING CONSIDERATIONS

Terrain drawing on the IRIS is accomplished in two different ways. The grid square method draws each terrain section as a square according to the resolution area. Each square is colored in a checkerboard motif. The actual terrain squares are drawn with the triangular mesh routine. The mesh method utilizes the triangular mesh routine to draw an entire row at each call. The triangular mesh routines

automatically connect an input point to the two previous points, creating a triangle in the mesh. The mesh method gives the terrain a smooth, realistic look to the terrain. Further, using the standard set up, the mesh routines draw the same terrain as the grid square method 3.1 frames per second faster with the average grid square value of 5.3 frames per second and mesh at 8.4 frames per second. Figure 9.1 shows the test results for four selected ranges and the two viewing angles. Figure 9.2 compares the advertised speed of 40,000 polygons per second against the speeds attained by the grid square and mesh methods. The major speed differences occur from the size of the polygons and the support code for moving the vessels around. The grid square method is left as a main run-time menu option for use in flat areas where the mesh method does not give any feeling for vessel motion.

D. TERRAIN LIGHTING MODEL

Lighting on the IRIS helps to increase the realism of three dimensional scenes. While lighting requires additional computation and memory allocation for the normals, an infinite light source with an infinite viewer is said, by the sales literature, to be computationally free. To test the effect of evaluating the lighting model on the overall performance, a toggle was placed on the run time menu that kept only the lighting model lmbind() call from being made. All terrain drawing routines were done in exactly the same way with the lighting model turned on or off. The picture with the lighting model off is nothing but an outline of the terrain, colored according to the last RGBcolor() calls. The frames per second meter showed an increase of one to almost two frames per second with the lighting model off. The number of polygons being drawn was constant at 1,800 polygons. While this again could be related to the

S&PV Performance Table

Draw Method	View Angle	View Distance	Polygons	Frames per Second	Polygons per Second
Grid Square	45	5,000	1,531	5.8	8,880
Mesh	45	5,000	1,531	10.0	15,310
Grid Square	15	5,000	863	10.6	9,148
Mesh	15	5,000	863	16.4	14,153
Grid Square	45	7,500	1,786	4.9	8,751
Mesh	45	7,500	1,786	9.6	17,146
Grid Square	15	7,500	947	9.9	9,375
Mesh	15	7,500	947	15.6	14,773
Grid Square	45	15,000	2,790	3.3	9,207
Mesh	45	15,000	2,790	5.9	16,461
Grid Square	15	15,000	1,333	7.2	9,598
Mesh	15	15,000	1,333	11.3	15,063
Grid Square	45	20,000	3,732	2.2	8,210
Mesh	45	20,000	3,732	4.3	16,047
Grid Square	15	20,000	1,675	5.3	8,878
Mesh	15	20,000	1,675	9.0	15,075

Figure 9.1 S&PV Performance Figures

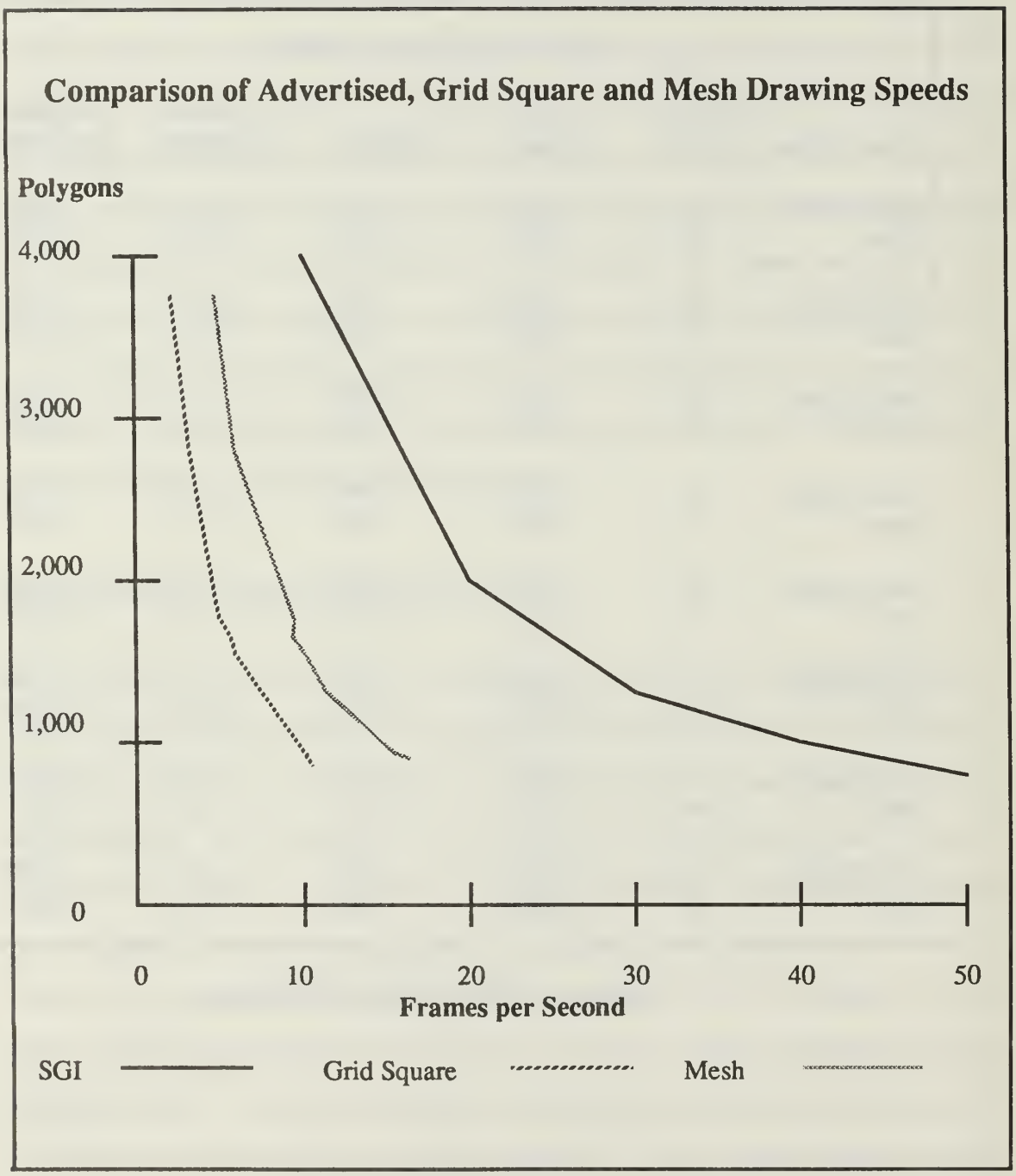


Figure 9.2 Performance Comparisons

memory shortage, it also highlights the computational expense of an effective lighting model such as used by the IRIS.

1. Comparison of the Lighted and Unlighted Versions

Another hint to the cost of an effective real time lighting model is in the performance differences found between the unlighted and lighted versions of the program. Before lighting was introduced to the S&PV, terrain drawing was done by color shading for distance. The terrain was initially colored a light green and was distance shaded to a dark brown Performance in this system for a 7,500 meter distance, 3,400 polygon picture was from 7-8 frames per second. The lighted version of S&PV in the same location, same distance and mesh draw on, only performs between 4-4.7 frames per second. Note, this test was done before a change was made to the control structure of the drawing algorithm speeding up the simulator to the values expressed in the earlier performance sections. Both versions of the program mentioned here had the same control structure at the time of the testing. Extra memory requirements and lighting model calculations make up the difference between the two program versions.

2. Normal Calculation Versus Normal Data Storage

In the third test, the terrain surface normals were calculated dynamically. The purpose of this test was to find the speed difference in calculating the normals against the necessary memory swap time. With the standard settings, the frames per second speed achieved was 3.2. While slower than the separate array storage method, it was consistent with the performance of the array of records storage scheme. This technique may become more practical in a multiple processor

environment where one processor handles the graphics pipeline and a second processor calculates the terrain surface normals.

E. PERFORMANCE CONCLUSIONS

Overall, the current drawing methods and storage scheme used in the program achieve the highest performance values of all the methods tested. With a multiple processor upgrade, such as the IRIS GTX, calculating the normals dynamically may become a viable option. The main performance constraint is still the lack of main memory. Upgrading the system memory beyond sixteen megabytes should allow greater performance with the program as it stands. Expanded memory will also allow for adding additional DMA data files to be brought into the program for use in the future surface and aerial views of the Command and Control Workstation of the Future.

X. CONCLUSIONS, LIMITATIONS AND FUTURE DIRECTIONS

The subsurface and periscope views system has fulfilled all of its initial design criteria as set out in the proposal starting the work. It is capable of accurately portraying the DMA Sea of Japan database along with easily recognizable three dimensional icons of surface and subsurface platforms in real-time. Two versions of terrain drawing are included in the package, the first for realism and speed, the second, for piloting on flat terrain. The current version of the S&PV system is an efficient simulator that can easily be modified to add whatever new features are desired.

A. CONCLUSIONS AND LIMITATIONS

1. Terrain Drawing

The terrain drawing algorithms within the S&PV system are fast and efficient. They represent terrain in an accurate, realistic way. While the simulator can be flown as an aircraft, modifications to the drawing routines should be made to draw only in low resolution, 400 yards by 400 yards, or in an even lower resolution. This would allow greater distances to be drawn without the severe performance penalties for large distances. Converting the drawing routines to run only in low resolution is a simple matter of setting the high and medium resolution boundaries to zero when in an aircraft mode.

The second area of concern with long distance drawing is the size of the database required to draw to the specified distance. Using a standard of visibility of

26 nautical miles, up to four database files may have to be loaded. The height data points would then account for nearly twelve megabytes of memory even before the terrain vertex normals are calculated. A compressed representation of the terrain height databases and the terrain vertex normals is necessary to perform the terrain drawing at this level.

2. Three Dimensional Vessel and Weapon Icons

The 3D vessel and weapon icons used in the S&PV are effective in portraying the desired platform. Currently only polygon normals are used on the icons but inclusion of vertex normals will allow for a more realistic, rounded shape to the icons. Creating the 3D icons is a time consuming task. Drawing tools should be created to remove the labor intensive aspects of these icons without sacrificing realism or performance.

While grounding detection has been implemented into the S&PV, collision detection with vessels and weapons has not. We are currently working on various schemes to efficiently detect collisions with complexity less than $O(n^2)$.

3. Networking

Networking between IRIS graphics workstations has been successfully integrated into the S&PV. Each workstation can generate targets to be viewed by all of the workstations on the network. There is no link between the S&PV and the earlier work on the Commander Display System [Ref. 1]. The Commander's Display System needs to be ported to the 4D series workstations and used as the vessel creation system to create a realistic, multistation simulation.

B. FUTURE DIRECTIONS

Future work on the S&PV should be directed in six basic areas, air and surface views, multistation networking with the Commander's Display System, 3D vessel and weapon icon construction, weapon's profiles, ship and submarine control surfaces, and sensor packages.

1. Air and Surface Views

The CCWF, with this work, has a realistic subsurface and periscope view ability. The air and surface views must be created. The current code will allow limited air and surface views, but the real problem lies in the amount of data necessary to provide terrain drawing out to the horizon, normally 26 nautical miles. Data structure techniques must be evaluated to determine the best way to store up to four sixty nautical mile database grids. Further work may also involve using the GTX upgrades ordered for two of the IRIS GT's. These upgrades provide multiprocessor support and if the processors can be divided into one working the graphics pipeline and the second working terrain vertex normal calculations, the database storage problem may be eliminated.

2. Multistation Networking

The Commander's Display System (CDS) was designed to display in color the Navy's tactical display system [Ref 1:pp 18-24]. The user could then select a vessel from his display and have the view displayed as appropriate. Currently, this is not possible. The CDS must be ported to a IRIS 4D system and updated. The networking within the CDS must be brought in line with the networking scheme used by the S&PV system.

3. 3D Vessel and Weapon Icon Production

Currently the S&PV has five vessel platforms and one weapon platform. Construction methods to produce these icons in a small percentage of the time currently taken must be done to allow the S&PV to effectively portray many possible vessel occurrences. As discussed in the chapter on 3D icons, work is being started in this area. For the S&PV to use the icons, the production systems should allow for scaling, in meters or yards, coloring and finally rotation. With the terrain data set up in one hundred yard squares with height in meters, the icons must be in the same scale as the terrain.

Coloring and shading is very important in recognition of vessels. The icon tool should allow the user to import the defined vessel colors from the simulation program. This will allow the tool to display the icon as it would be seen from the simulator.

The S&PV assumes all vessels and weapons are oriented to mathematical zero, 090 true, degrees. The display system is then set to automatically rotate and display the vessel, wreck or weapon to its course and position. Icons created by the icon tool must allow the icons to be centered at whatever position is desired.

4. Weapons Profiles

While creating 3D icons for weapons is a fairly easy task, creating the weapons profiles will require a fairly major effort. Modern weapons contain intelligence that guides them to their target. While all are initially guided by the vessel platform they were launched from, many have the ability to deviate from the

initial launch programming and seek out its victim. All of this makes the presentation of dumb weapons obsolete.

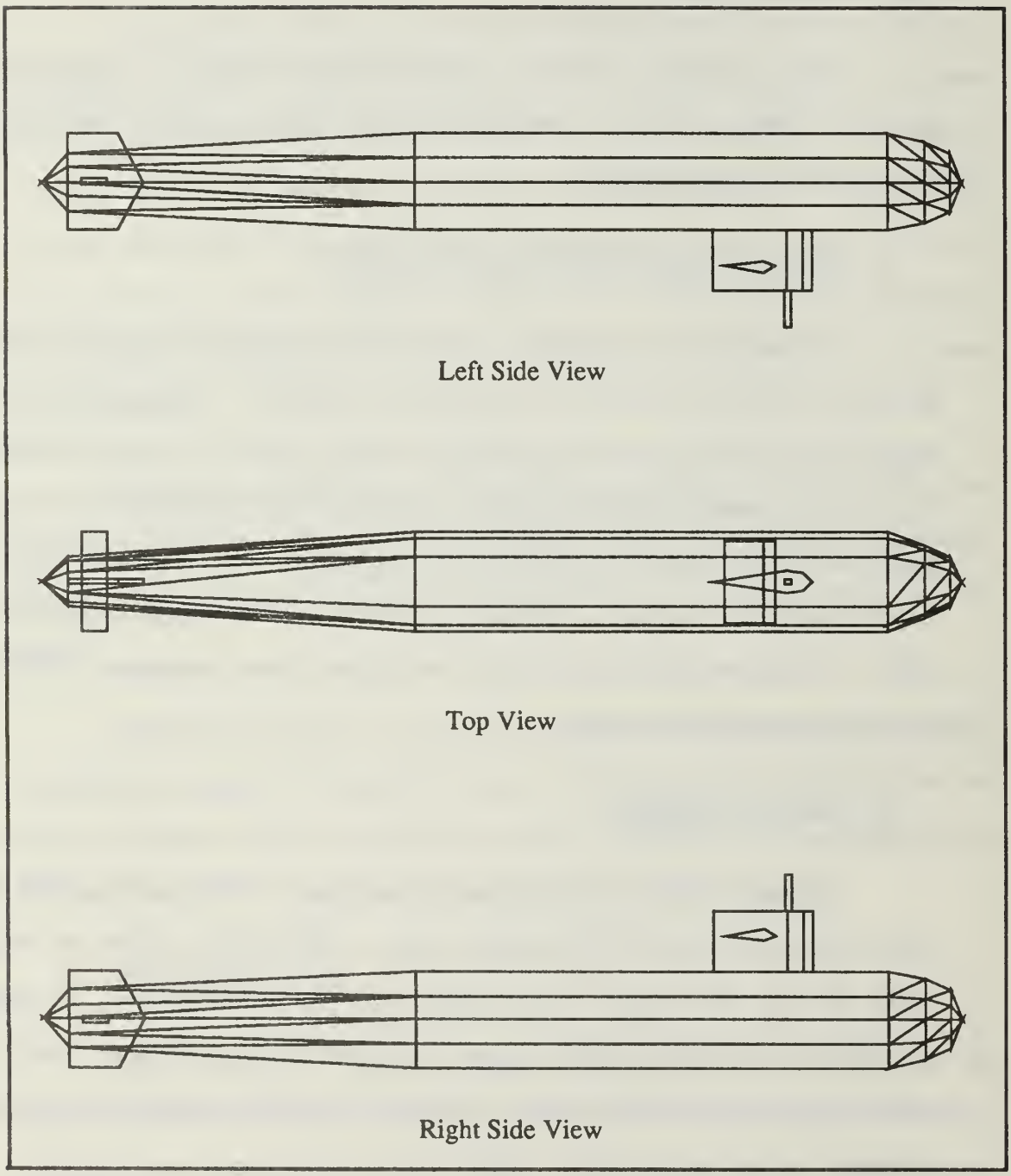
Weapon storage within the CCWF allows for inclusion of a path file or pointer to a guidance routine. This ability allows the CCWF to advance further into a fully realistic training simulation.

5. Ship and Submarine Control Surfaces

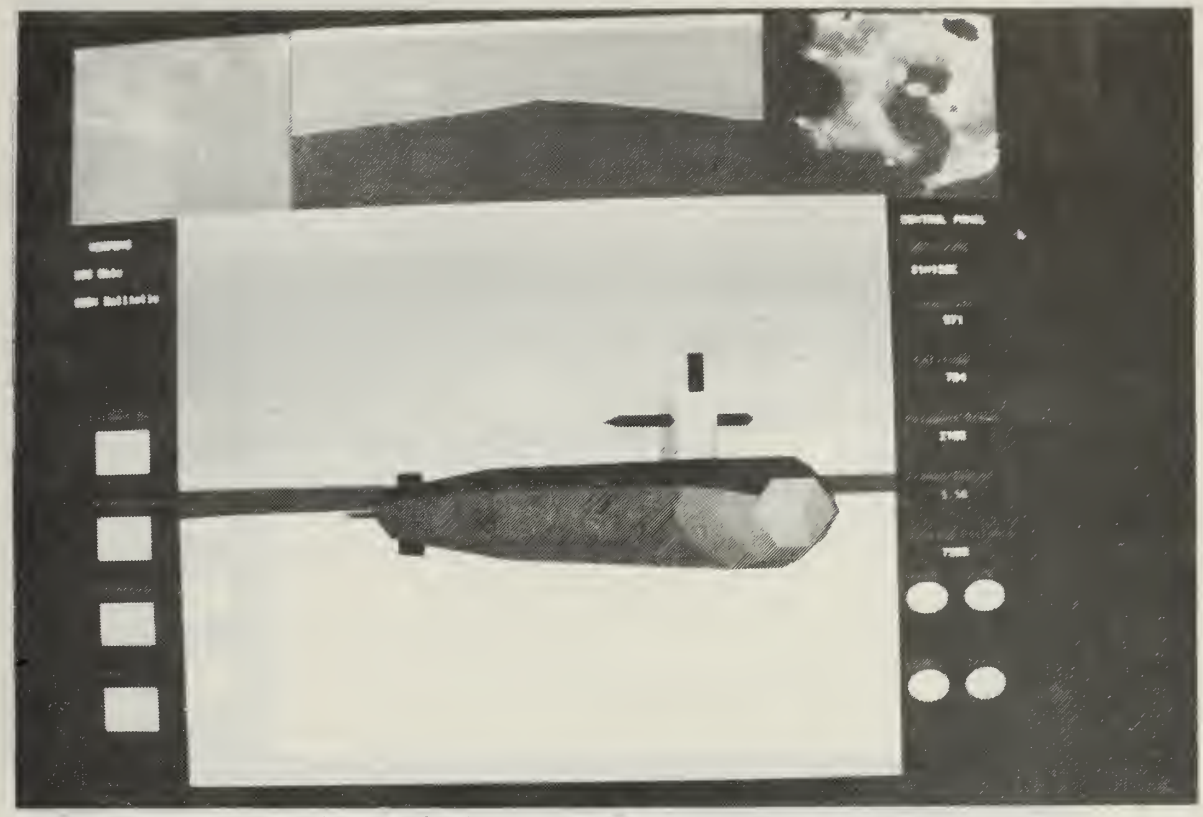
To further add to the realism of the S&PV, vessel driving techniques should be altered to reflect the vessel's actual operating parameters. Instead of a dial for changing the course, the dial should change the rudder angle of the ship or submarine. The depth dial should change the setting on the dive planes of the submarine and the speed dial should respond as an engine order telegraph. This requires the ability to derive realistic and unclassified operating parameters for the vessels included in the S&PV. Maneuvering through channels and around vessel formations would then become a realistic training scenario.

6. Sensor Packages

Modern vessels are packed with sensing gear. From sonars to ESM gear, they are the eyes and ears of the modern warship. While adding realism and training value, this step will assuredly put the CCWF into the classified arena. Regardless, as much of the sensor packages as can be allowed without having to classify the S&PV should be added to the program in increase the training and realism level of the program.

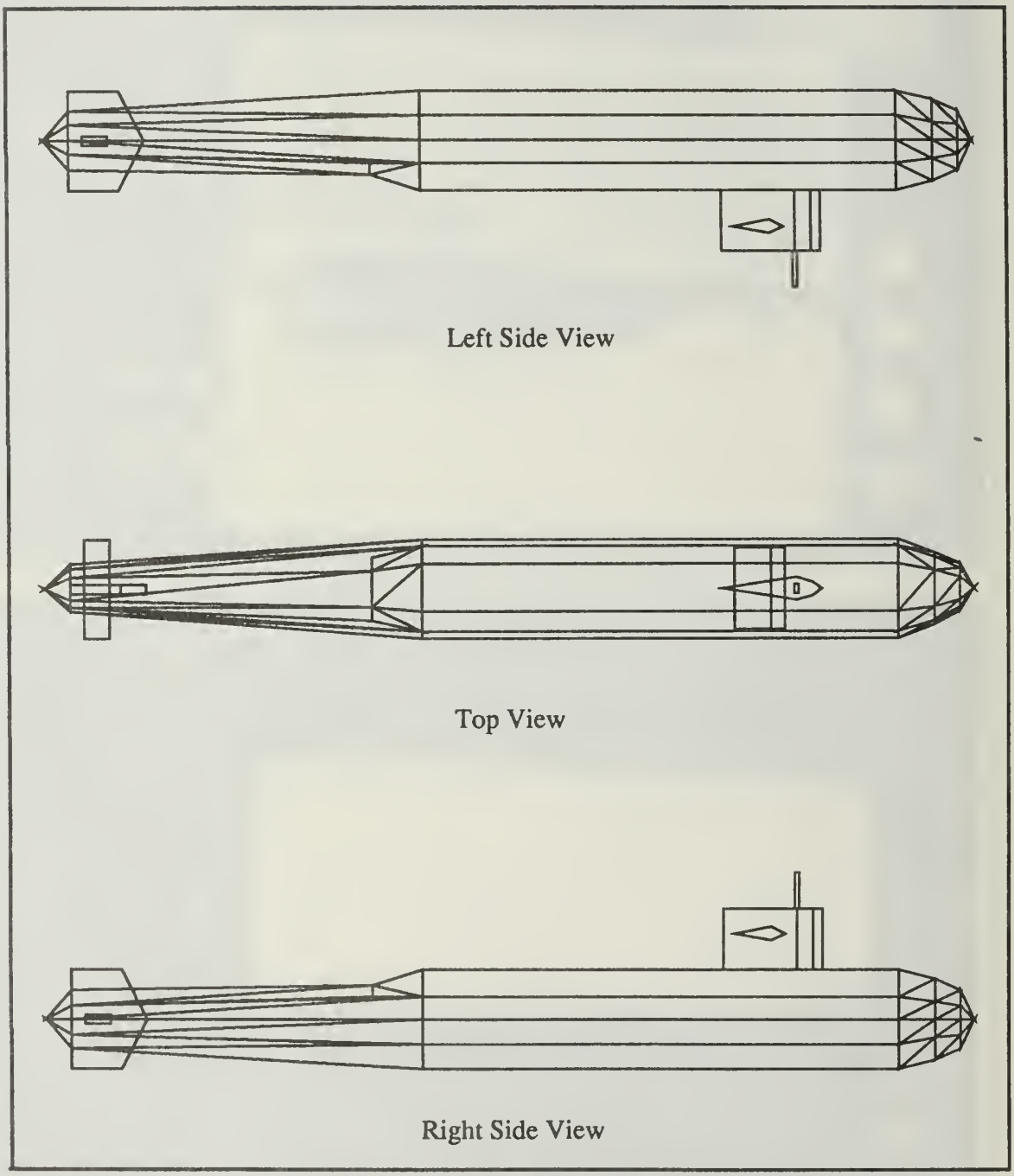


Appendix A.1 Los Angeles Class Submarine

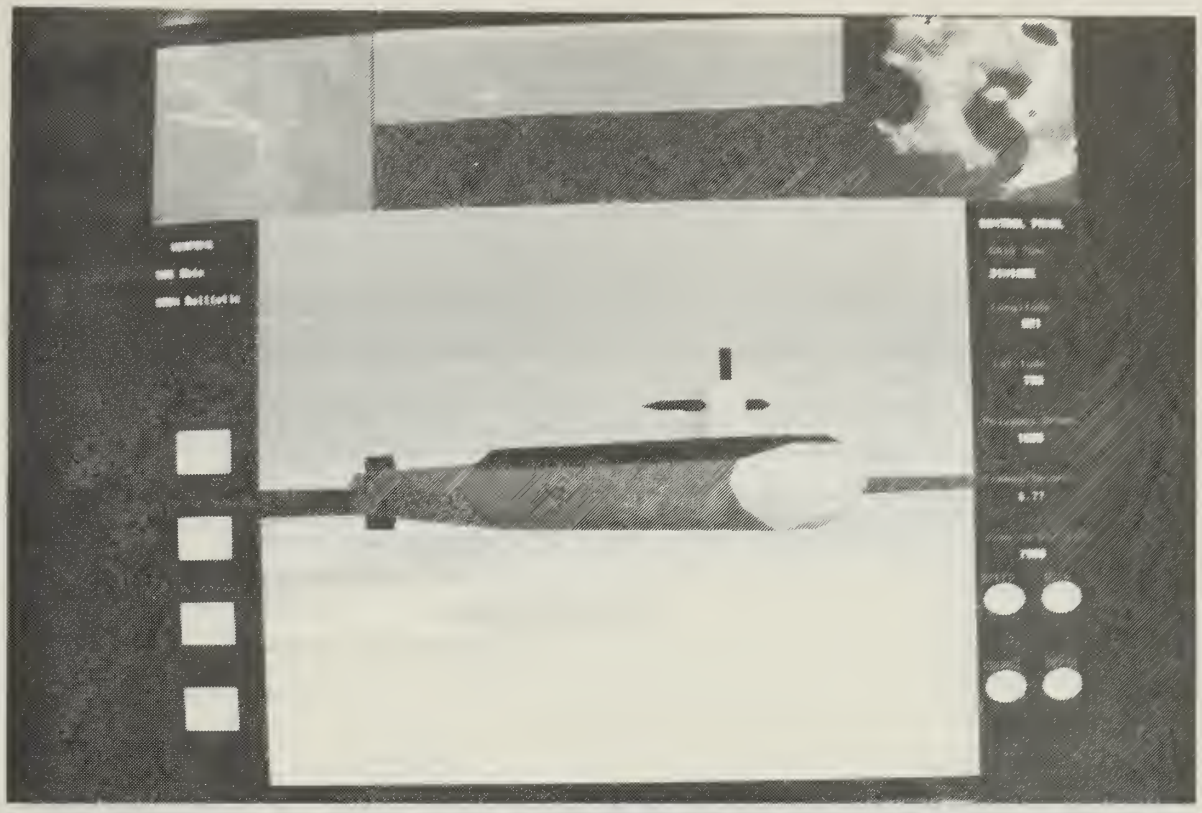


Appendix A.2 Los Angeles Subsurface View

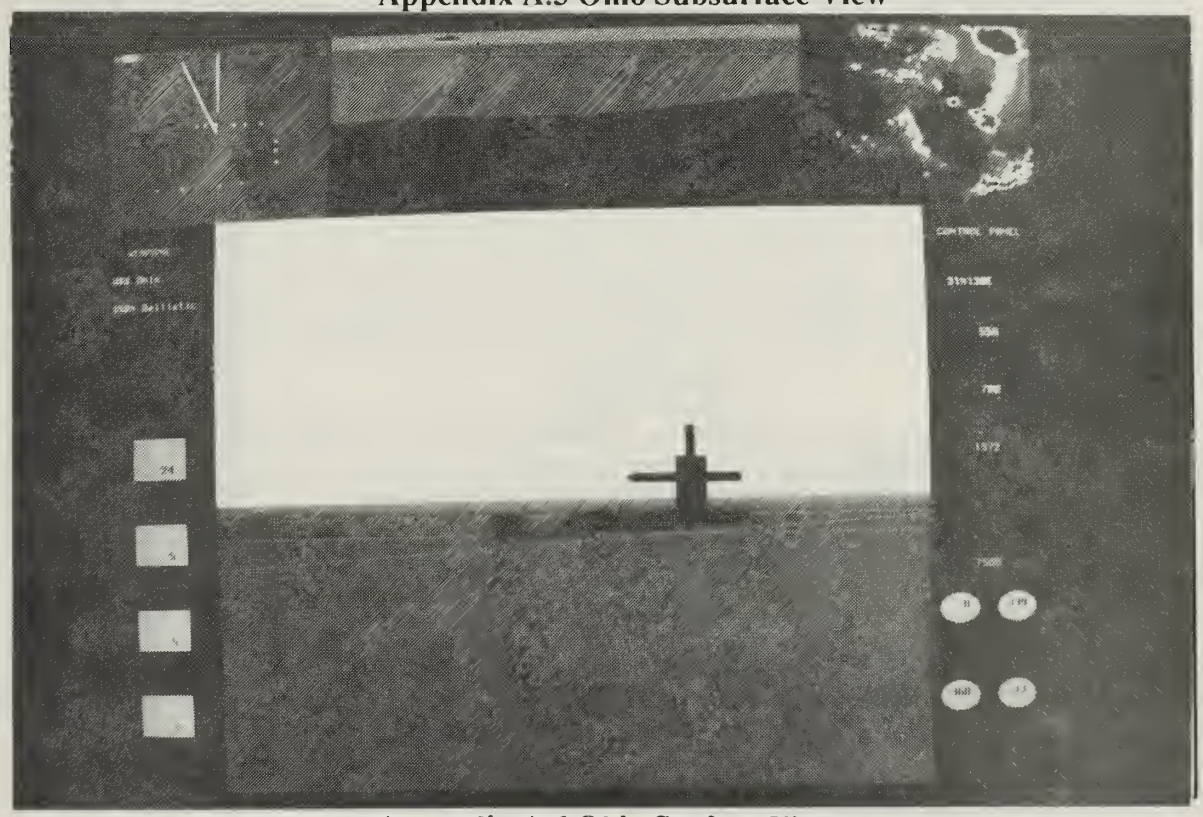
Appendix A.3 Los Angeles Surface View



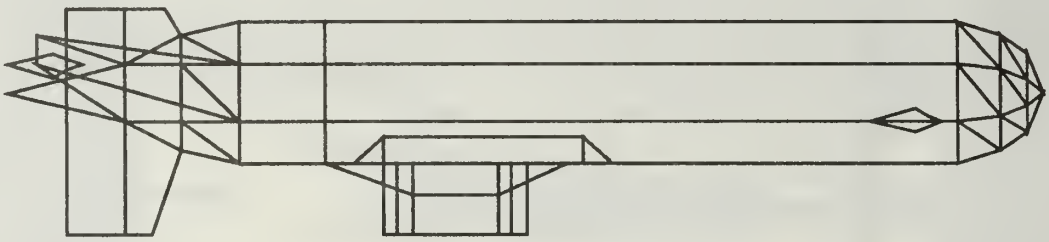
Appendix A.2 Ohio Class Submarine



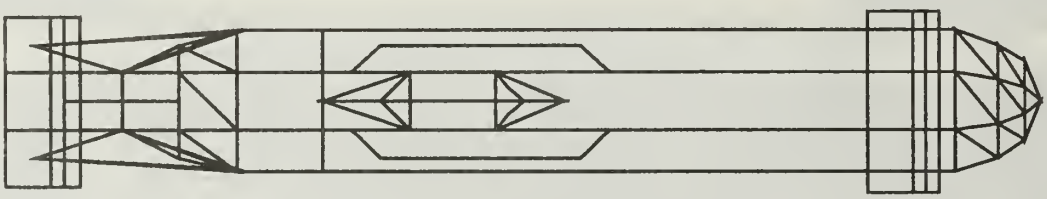
Appendix A.5 Ohio Subsurface View



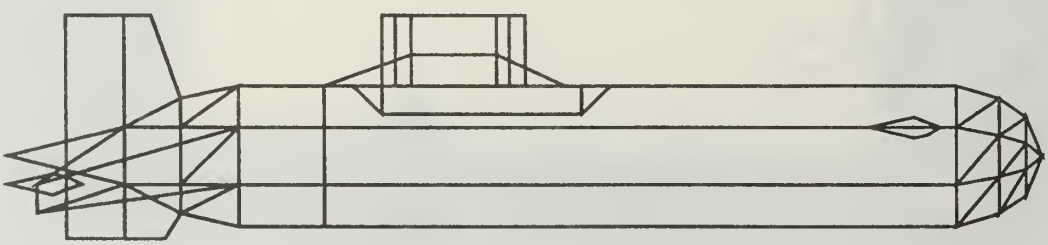
Appendix A.6 Ohio Surface View



Left Side View

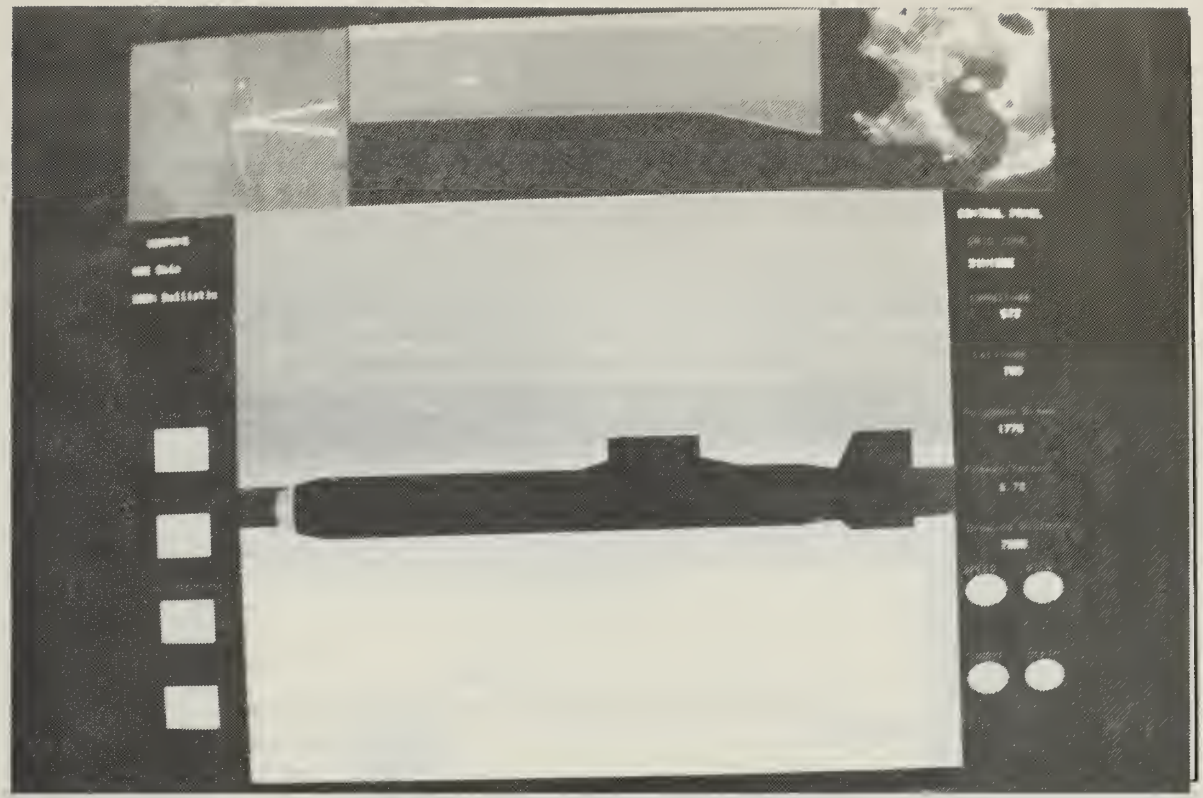


Top View

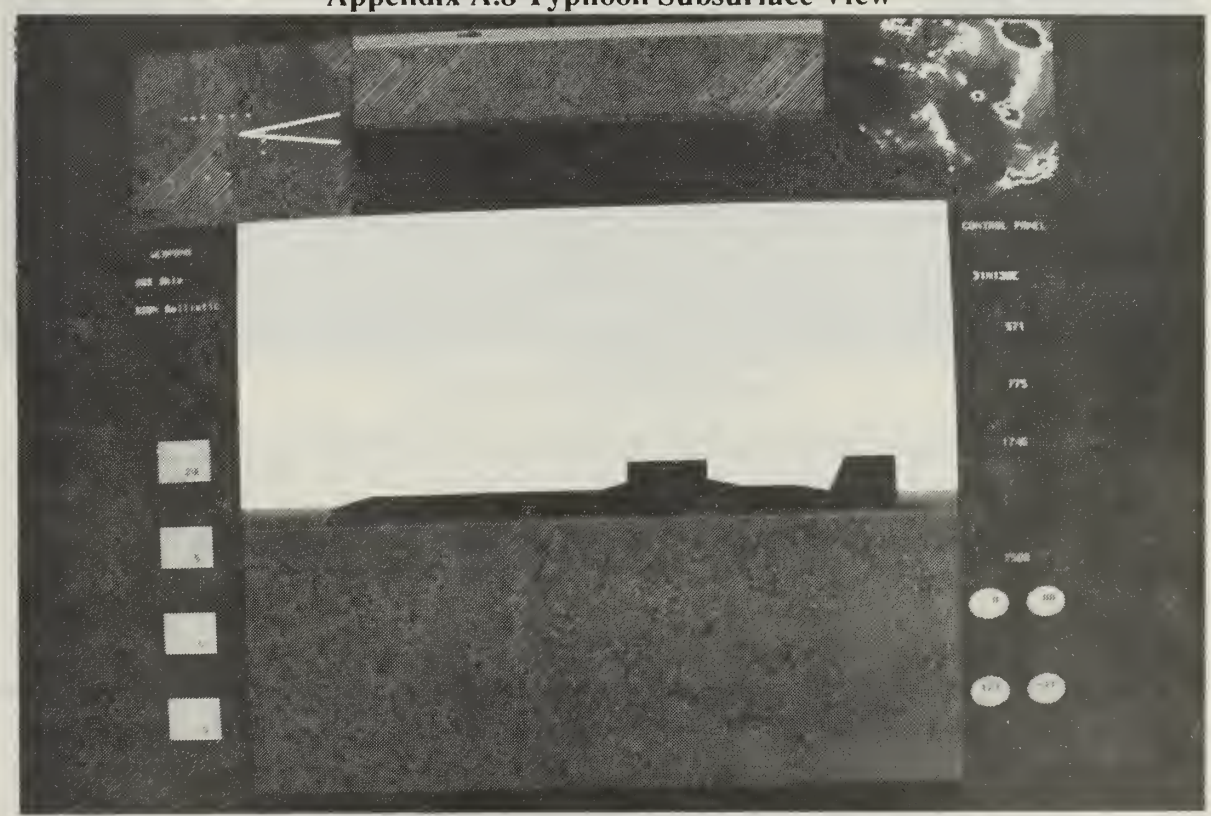


Right Side View

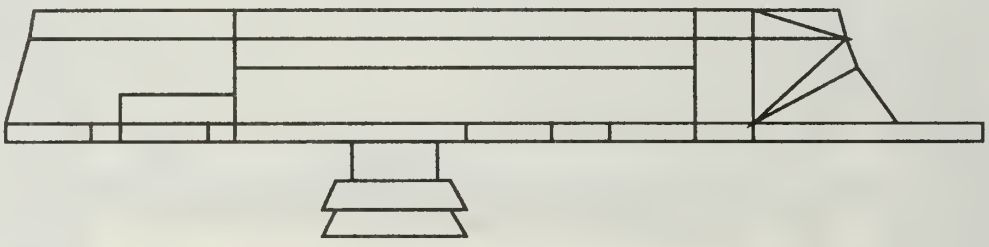
Appendix A.7 Soviet Typhoon Class Submarine



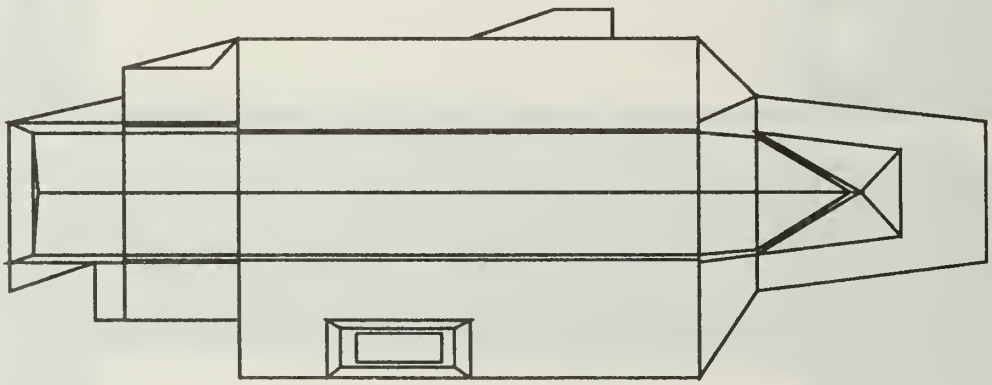
Appendix A.8 Typhoon Subsurface View



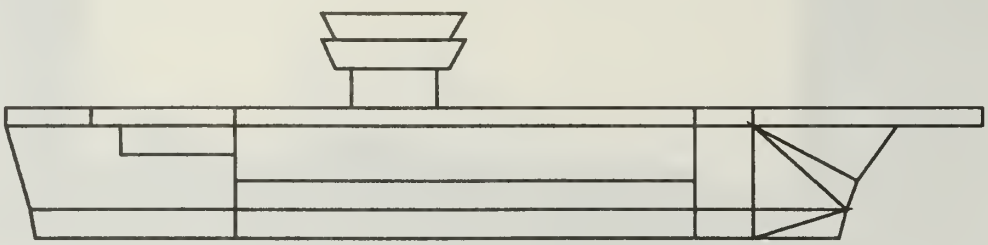
Appendix A.9 Typhoon Surface View



Left Side View

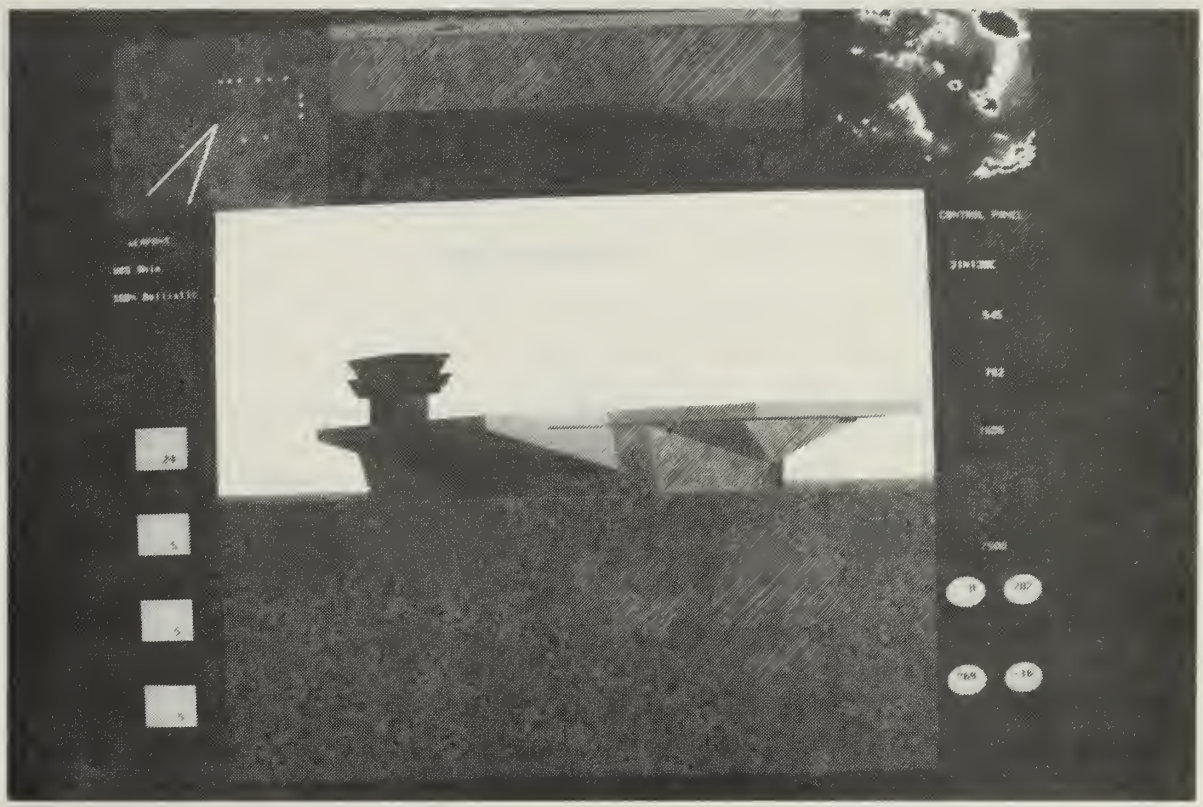


Top View With Hull Lines

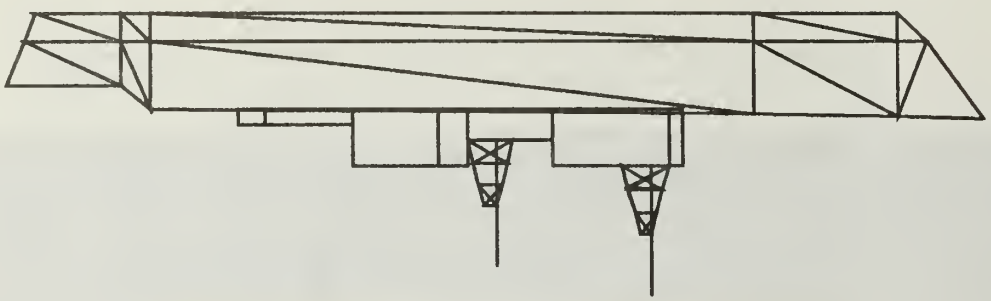


Right Side View

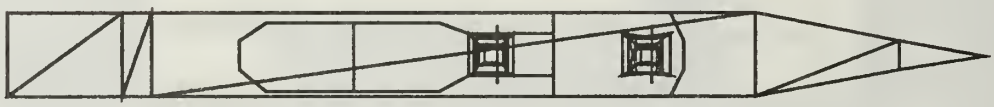
Appendix A.10 Kitty Hawk Class Aircraft Carrier



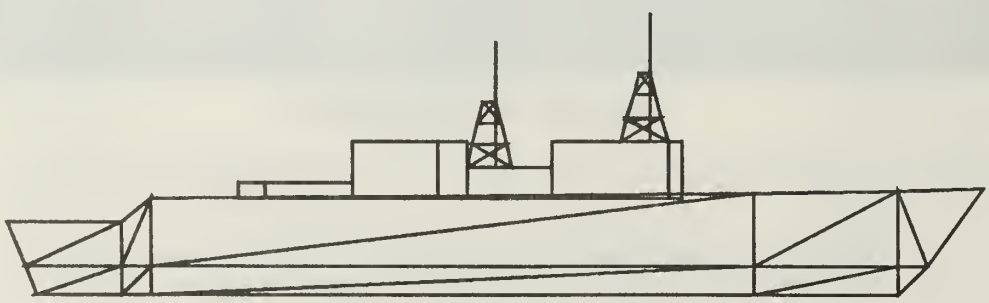
Appendix A.11 Kitty Hawk



Left Side View

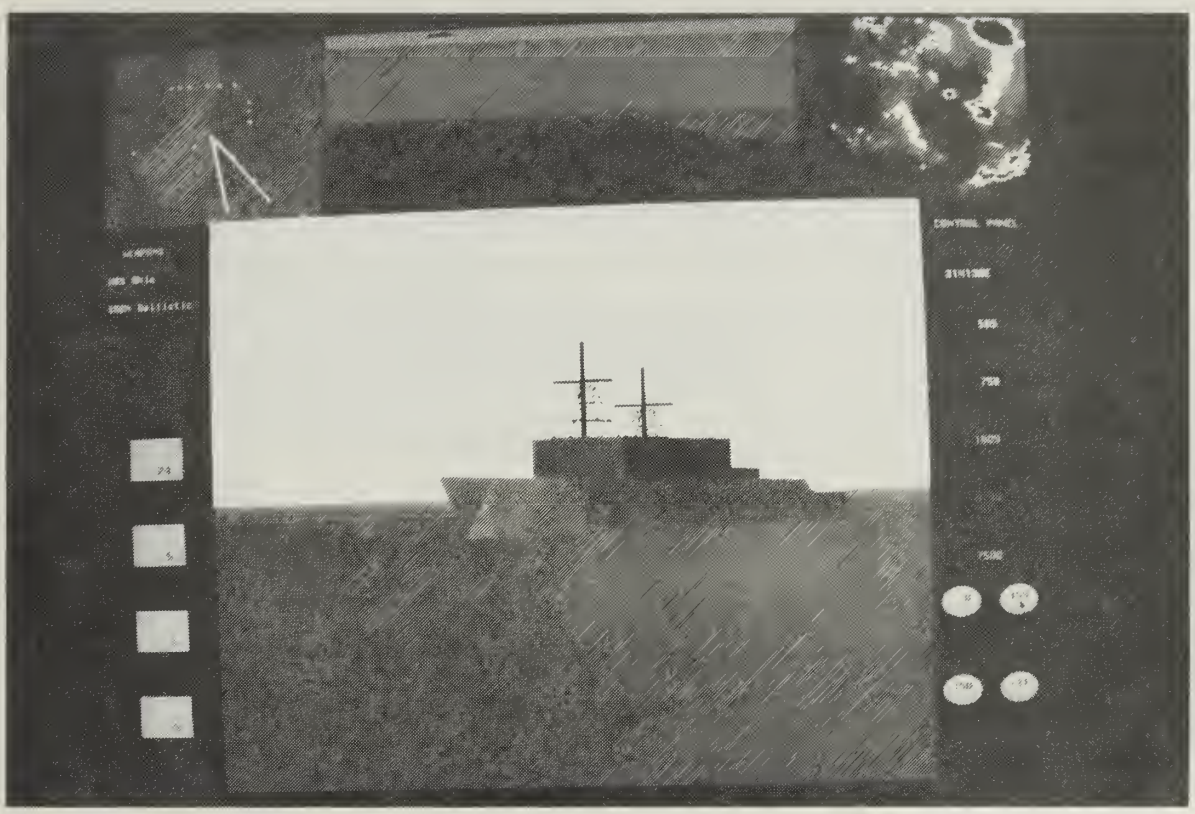


Top View

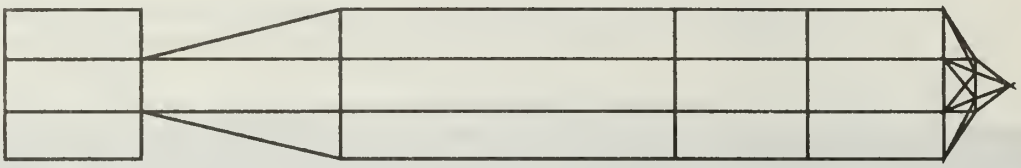


Right Side View

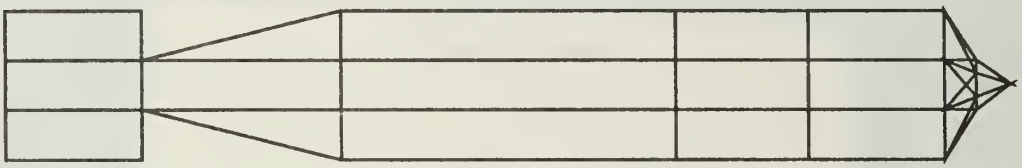
Appendix A.12 Spruance Class Destroyer



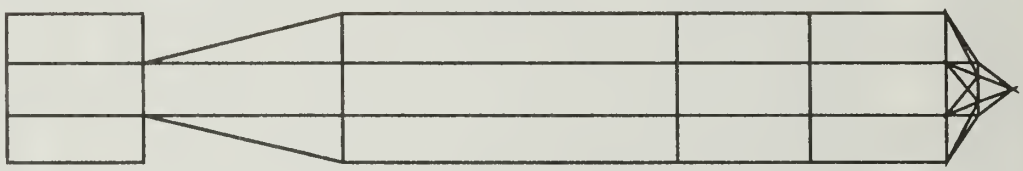
Appendix A.13 Spruance



Left Side View



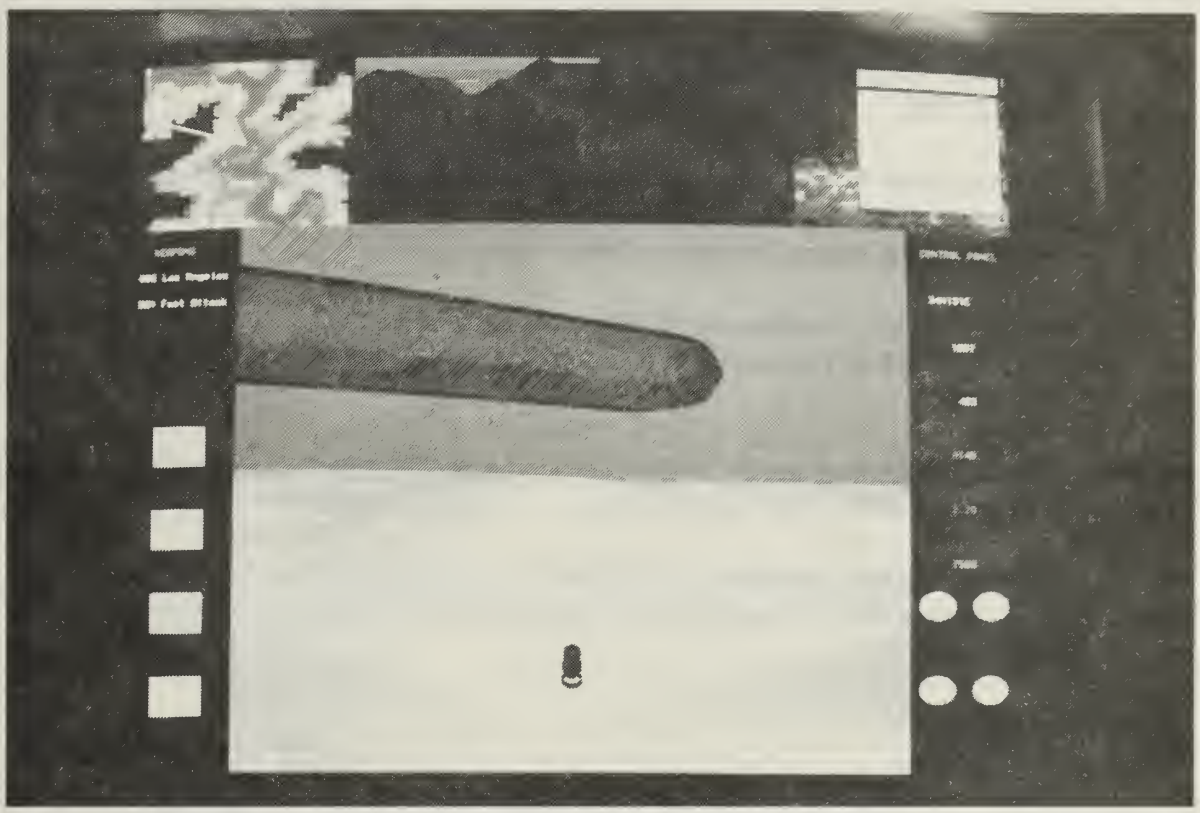
Top View



Right Side View

Bottom View

Appendix A.14 Torpedo



Appendix A.15 Twenty-one Inch Torpedo

LIST OF REFERENCES

1. Adams, Rodney M., *A Software Architecture for a Commander's Display System*, M.S. Thesis, Naval Postgraduate School, Monterey, California, April 1987.

2. Akeley, Kurt and Jermoluk, Tom, "High-Performance Polygon Rendering," *Computer Graphics*, SIGGRAPH 1988 Conference Proceedings, v. 22, no. 4, August 1988.

3. Fichten, Mark A. and Jennings, David H., *Meaningful Real-Time Graphics Workstation Performance Measurements*, M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1988.

4. Harris, Frank E., *Preliminary Work on the Command and Control Workstation of the Future*, M.S. Thesis, Naval Postgraduate School, Monterey, California, June 1988.

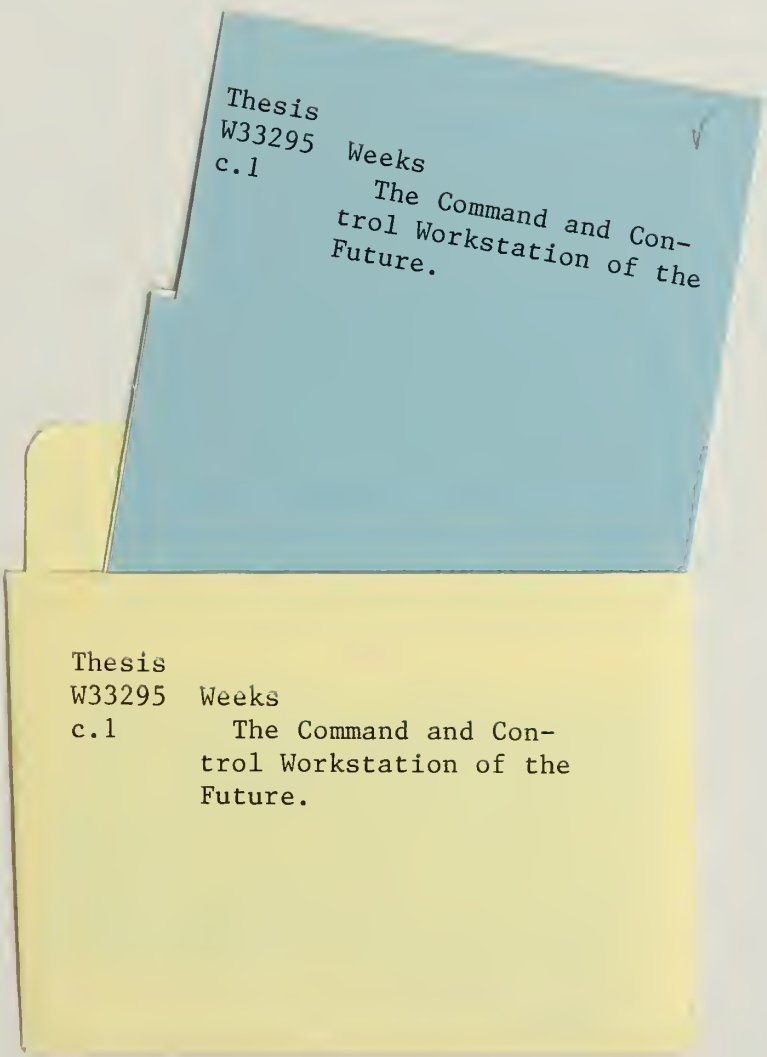
5. Hearn, Donald and Baker, M. Pauline, *Computer Graphics*, Prentice Hall, Inc., Englewood cliffs, New Jersey, 1986.

6. Olivera, Hector J. Mariscal, *EDITFONT - An Interactive Font Editing System*, M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1987.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Dr. Michael J. Zyda Naval Postgraduate School Code 52, Department of Computer Science Monterey, CA 93943-5100	200
4.	CPT Charles E. Phillips 555 Creek Road Poughkeepsie, NY 12601	2
5.	LT Gordon K. Weeks Jr. 1039 Sycamore Rd. Graham, NC 27253	2
6.	John Maynard Naval Ocean Systems Center Code 402 San Diego, CA 92152	1
7.	Duane Gomez Naval Ocean Systems Center Code 433 San Diego, CA 92152	1
8.	James R. Louder, Naval Underwater Systems Center Combat Control Systems Department Building 1171/1, Newport, RI 02841	1

9. Research Administration, Code 012 1
Naval Postgraduate School
Monterey, CA 93943-5000



Thesis
W33295
c.1
Weeks
The Command and Control Workstation of the Future.